



[HTTP://www.dotnetrocks.com](http://www.dotnetrocks.com)



Carl Franklin

Carl Franklin and Richard Campbell interview experts to bring you insights into .NET technology and the state of software development. More than just a dry interview show, we have fun! Original Music! Prizes! Check out what you've been missing!



Richard Campbell

Text Transcript of Show #482
(Transcription services provided by [PWOP Productions](#))



Leon Gersing is Having a Love Affair with Ruby!

September 17, 2009

Our Sponsors



Developer
EXPRESS

[HTTP://www.devexpress.com](http://www.devexpress.com)



CoDe

component developer magazine

[HTTP://www.code-magazine.com](http://www.code-magazine.com)



[HTTP://www.telerik.com/](http://www.telerik.com/)



Leon Gersing is Having a Love Affair with Ruby!
September 17, 2009

Geoff Maciolek: The opinions and viewpoints expressed in .NET Rocks! are not necessarily those of its sponsors, or of Microsoft Corporation, its partners, or employees. .NET Rocks! is a production of Franklins.NET, which is solely responsible for its content. Franklins.NET - Training Developers to Work Smarter.

[Music]

Lawrence Ryan: Hey, Rock heads! Put down that Ruby doobie and listen up! It's time for another stellar episode of .NET Rocks! the Internet audio talk show for .NET developers, with Carl Franklin and Richard Campbell. This is Lawrence Ryan announcing show #482 with guest Leon Gersing, recorded live Monday, August 24, 2009. .NET Rocks! is brought to you by Franklins.NET - Training Developers to Work Smarter and now offering SharePoint 2007 video training with Sahil Malik on DVD, dnrTV style, order your copy now at www.franklins.net. Support is also provided by Telerik, combining the best in Windows Forms and ASP.NET controls with first class customer service, online at www.telerik.com, and by Red Gate Software, essential tools for SQL Server, .NET, and Exchange. Support is also provided by CoDe Magazine, the leading independent magazine for .NET developers, online at www.code-magazine.com. And now, the man who thinks Jimmy Huff is hiding out on Leon's beard, Carl Franklin.

Carl Franklin: Thank you very much and welcome back to .NET Rocks! Carl and Richard here for you for the next hour or so. Hey, Richard.

Richard Campbell: Howdy, sir. How are you?

Carl Franklin: Good. Now that that's out of the way, let's get into Better...

Richard Campbell: Oh yeah, the quick intro. Let's go.

Carl Franklin: Now that that's out of the way, let's get into Better Know a Framework.

[Music]

Carl Franklin: I don't know, I'm just feeling very -- I'm feeling the pressure from Scott Hanselman who's like, "Don't waste my time."

Richard Campbell: "Don't waste my time, go faster."

Carl Franklin: "Go faster."

Richard Campbell: I've got an email just like that for you today.

Carl Franklin: I want to listen to this thing on speed read and knock your board, okay. So on Better Know a Framework today we're going to talk about System.Windows.Media.TextFormatting which is where all the types are, the control formatting of text in WPF typically at a lower level than the control-based text presentation or the text object model.

Richard Campbell: Huh.

Carl Franklin: Isn't that cool? Not really, not really but you've got to know it's there.

Richard Campbell: You've got to know it's there, you've got it in the format text.

Carl Franklin: You've got TextEmbeddedObjects, you have TextEndOfLine, you have TextFormatter, TextHidden, TextLine, TextLineBreak, Text this, Text that, Text the other thing, go check it out yourself. There's one interesting one in here, TextRun, which represents a sequence of characters that share a single property set.

Richard Campbell: Huh.

Carl Franklin: Isn't that interesting?

Richard Campbell: It is interesting, yeah.

Carl Franklin: So I'm thinking of this for formatting if you've got a lot of things that all have the same property sets, so any change to the format like font style, foreground color breaks the TextRun.

Richard Campbell: Right.

Carl Franklin: There's an example of how to change formatting into TextString that results in a series of TextRuns and each has a common set of formatting properties. So it's a way of grouping like objects together according to their format.

Richard Campbell: Cool.

Carl Franklin: So Richard, got an email? Quick, hurry.

Richard Campbell: In the spirit of Scott Hanselman, get ready for this.

Carl Franklin: Go, go, go.

Richard Campbell: "Hi, Carl and Richard. I just finished listening to that latest episode of .NET Rocks! with Sahil Malik. It's one of the best episodes ever. It's both informative and entertaining. It was a video, it was a recording. Yeah, we started a large



Leon Gersing is Having a Love Affair with Ruby! September 17, 2009

SharePoint project last year. Keep up the good work. Vladdick."

Carl Franklin: I love it.

Richard Campbell: That's a good email. Thank you, Vlad. If you've got questions, concerns, ideas, criticisms, send us an email, dotnetrocks@franklins.net.

Carl Franklin: And with that let's introduce our friend, Leon Gersing. Leon has been bringing value to clients large and small for over ten years and has a passion for technology, art and community. He has worked professionally on the .NET platform since its release and recently joined EdgeCase in Columbus, Ohio with a focus on building agile solutions using Ruby and Ruby on Rails. A believer in building strong communities, Leon spends time presenting on a wide variety of development topics at events and user groups in the region. He loves nothing more than to be around other developers working together to create something unique and fresh, something that has never been done before. He believes there is no challenge that can't be overcome with passion and creativity. He spends his spare time with his wife, two beautiful girls, two sweet kitties, and his ukulele. Welcome, Leon.

Leon Gersing: Thanks guys. What's up?

Carl Franklin: So the troublemaker of DevLink.

Leon Gersing: I'm the troublemaker of just about any party I go to.

Richard Campbell: Yeah, yeah. You're a troublemaker at CodeMash too as I recall.

Carl Franklin: That's right.

Leon Gersing: Yeah. I mean, any time I can stir the pot, I might as well do it.

Richard Campbell: I'll also give you this complement. I was headed up to the open space's section at DevLink, I had to step around Leon's holding court. There must have been 30 guys sitting in the hallway in a ring around Leon.

Carl Franklin: That's great.

Richard Campbell: I don't even know what you guys are arguing about, but you were obviously winning.

Leon Gersing: Well, we weren't actually arguing. A couple of guys had asked if I would give them a Rails from the ground up kind of tutorial.

Richard Campbell: Wow.

Leon Gersing: And I said, "All right. Grab your laptops," and really it started with three of us and then somebody would walk by and say, "Hey, what are you guys doing?" And I would say, "Well, we're learning Ruby on Rails from the ground up." And they wanted to see the No Fluff, stuff. They wanted to know exactly what was being generated, exactly what's in the framework, and exactly how they can leverage it.

Richard Campbell: Right.

Leon Gersing: So that's what we did.

Richard Campbell: You basically did an impromptu session.

Leon Gersing: Yeah. It certainly lasted as long as the morning session I did on the con. It was, I think, three hours that we're in the hallway.

Richard Campbell: Holy cow.

Leon Gersing: Yeah. I was pretty tired at the end. I was like, "Guys, I need a drink." "All right, come on."

Richard Campbell: You needed something.

Leon Gersing: Yeah.

Carl Franklin: So we invited you here because during the DevLink panel show a few weeks ago you came up to the microphone, and the topic was "Is software development too complex?" and of course we had a lot of .NET people on the panel, but we did at the beginning of the show opened it up to discussion to software in general.

Leon Gersing: Yeah.

Carl Franklin: You know, I don't want to focus on .NET in particular and you stepped up at the microphone and said, ".NET is complex, Ruby is great, everything else sucks. Read about it," and turned around and walked away.

Leon Gersing: Oh yeah.

Carl Franklin: You know, I'm paraphrasing.

Leon Gersing: I'm sure that's what everyone heard.

Carl Franklin: I'm paraphrasing but...

Richard Campbell: I think somewhere in there you also said none of you are qualified to comment on Ruby, too.

Carl Franklin: Yeah, I think you're right.

Leon Gersing: If I remember right, and I haven't listened back to that show yet, I don't know if it has been released at the time of this recording, I think Carl, you brought it up. You said, "Well, is Ruby going to save us all?" or something. It's was very like -- I don't want to say flip, but it was a little flip.

Carl Franklin: Absolutely. That's my job, to be flip.

Leon Gersing: Yeah, exactly, stood apart.

Carl Franklin: Stood apart.

Leon Gersing: You know, poke the bear. And I saw two or three people with very little Ruby experience immediately turn it off, immediately shut down and that flies on the face of everything I think software craftsmen, and developers, and architects should be doing when they evaluate any solution. You should never just say, "Oh well, no, of course not. That won't work for me." You've got to look into it. You have to understand it, and I'd happened to be in a couple of open spaces with some of those people, not all, but some who were saying those things flat out admitted that they knew nothing of Ruby or what it had to offer, yet felt free and confident to sit on a panel and say it has no value to them. I think that's a dangerous road that we can go on and if we're looking to this panel of experts, of people that should be our thought leaders, I think that they should set a better example.

Carl Franklin: Yeah. Well, also to be fair, you know when you approach any panel or you evaluate it, you have to take the expertise of the panelist at face value. I mean, and I think that's probably what you were just saying, it's that when we have a panel, we can't obviously get a representative of every technology out there.

Leon Gersing: Of course.

Carl Franklin: So you have to understand that these were .NET guys...

Leon Gersing: Of course, and if I were sitting on a panel and somebody asked me to evaluate Spring's relevance on Java I would probably say something to the effect of I'm not familiar with it.

Carl Franklin: I'm not familiar with it, yeah.

Richard Campbell: I'm not the guy.

Carl Franklin: Good point.

Leon Gersing: I certainly wouldn't go, "Well, Spring is completely irrelevant," because I don't know. Maybe it is, maybe it's not.

Carl Franklin: I don't recall anyone saying that, but I do recall...

Leon Gersing: No, no, no, no, it wasn't that.

Carl Franklin: I do recall it being dismissed, yeah.

Leon Gersing: But yes, it did feel dismiss and that was my only point and I didn't want to start a flame where I was just saying I don't believe that you can universally dismiss what I believe to be the right kind of abstraction language level instead of building more and more products to make other languages do things that they're not inherently able to do.

Carl Franklin: Well, let's finish that discussion because the panel topic was has software development gotten too complex, and you basically said .NET is complex, everything else is easy.

Leon Gersing: I wouldn't say everything else.

Carl Franklin: That's what you said though.

Leon Gersing: But I would say that there are certainly alternatives that are easier.

Carl Franklin: That's what you said.

Leon Gersing: I did not say "everything else," right.

Carl Franklin: You said "everything else is not."

Leon Gersing: Oh, okay. Well, let me take that back then if that's exactly what I said.

Carl Franklin: Because I'm looking at shrinkster.com/18x right now which lists PHP MVC Frameworks, MVC Frameworks written in PHP, and there are 71 of them.

Leon Gersing: Sure.

Carl Franklin: So, you know.

Leon Gersing: I don't know the complexity of those in particular, and I mean is multiple choice analogous to complexity?

Carl Franklin: No, but it certainly makes the -- that was one of the topics on the panel, it was too much technology, so many options, so many ways to do the same thing because Microsoft is like a software factory and they just churn out these solutions that first of all how do you know what to pick, and second of all when you go to sit down and install it, how do you know what's the current version of anything because try to find the current version of anything on the internet is really difficult so the barrier to entry is complex just because of the sheer volume of options that you have.

Leon Gersing: Sure, and that's to be certain, but when you start narrowing down your specific needs then that window gets significantly smaller. Now you may not want to say -- so you may say, all right, I'm starting with the Web app and I'm on the .NET platform so I've got two choices now.

Carl Franklin: Yeah.

Leon Gersing: Well, you've got more than two choices and two choices are pretty easy to evaluate. I don't think anybody is going to argue that.

Carl Franklin: Right. Well, yeah. I mean especially, I think, where you get into a lot of issues is the whole data layer thing. I mean, data access technologies are gone, and not only data access but architectural tools. I mean, you know, whether to use an ORM or not. Which one, what flavor, code generation? I mean, there are just so many options and ways to do it and probably no matter what platform you're on you have these kinds of options.

Leon Gersing: Sure but I don't see choice as a barrier. I don't think choice is analogous to complexity in any way.

Carl Franklin: Not by itself but when you add on top of that CTP's and beta 1's and SP1's and SP2's and got to get this version, oh but that page, that version had been updated so the page that you're reading that says you've got to get this one is no longer valid but it's still out there, you know, that kind of stuff.

Leon Gersing: Well, your vendor, if you're in a vendor specific area they'll tell you what you're supposed to have, and if you're behind a firewall then you probably have something on your ELA that says what version you can actually accept so that's going to limit your choices too. If you're starting from greenfield you're going to look at choices that are basically going to give you the lowest cost of entry.

Carl Franklin: Yeah.

Leon Gersing: So you want to load TSE and a higher ROI and those things, I think, are a little easier to see from 50,000 feet up.

Carl Franklin: Yeah and I'm speaking purely about .NET because that's been my experience and the complexity is higher because of the barrier in entry, because of this constant releases of things not knowing what...

Leon Gersing: Sure.

Carl Franklin: You know, it's great when you can just take a product off the shelf to install. One install, put it all in there and you have all the choices that you need in check boxes right there in one product and it doesn't require any external things.

Leon Gersing: Yeah. I would be the opposite. I like to pull from a lot of different sources. I like to configure as I'd like, and I want the flexibility to open it up after I've installed it and see if maybe the internals are something I want to shake around because when I get things like version numbers that I can use, if there's something wrong in there then I'm stuck with it.

Carl Franklin: Yeah.

Leon Gersing: So I used to do SharePoint consulting like three years ago and we do this custom SharePoint code, and that version of the API was busted. So there were calls that were -- the APIs themselves were literally broken, they didn't work. So I would go and make a patch to connect.microsoft.com or whatever, and I'd say great, we got it, we submitted it as a bug, we'll let you know when it's release. I never got into another service stack, I never heard anything about it. I mean it never, ever worked again. Again, I haven't looked in the last year but for a long time it didn't work.

Carl Franklin: Yeah.

Leon Gersing: For me that limits my ability to reduce the complexity of my projects. I can't depend on something that's basically broken.

Richard Campbell: Well, that's a total vendor lock-in you had there and there's only one source to be fixed. You have no alternatives.

Leon Gersing: That's absolutely right. To me that's a big negative when taking any product.

Carl Franklin: That really sucks and it's been in this business forever. I mean, use any third party tool, you're beholden to them.

Leon Gersing: Sure.

Carl Franklin: Yeah.

Leon Gersing: Absolutely.

Richard Campbell: We definitely deal with folks out there who are totally fearful of third party tools of any kind because the moment Microsoft moves and make a new version of anything, you're now stuck waiting. I can't upgrade until these four other companies get around to upgrading too.

Leon Gersing: Hmm, and then you end up with this great not invented here mentality where you have to actually build everything that you see outside of your sandbox that looks cool. You have to build it internally, which usually given the scope and time, is far inferior to the product that you wish you could use.

Richard Campbell: Yeah and you'll never get it better. That builds its own problem which means I can't upgrade my app now because there's so much code to be written.

Leon Gersing: Right, exactly. Yeah, that's a problem. That's definitely a problem.

Carl Franklin: So tell us about Ruby. Tell us about your experiences with Ruby.

Leon Gersing: Where to start. I'm currently having a love affair with Ruby. I've been having it for, I don't know, three or four years now mainly because of I would say my experience as an ASP.NET developer. There were a lot of things that I wish I had, I wish I could do, wanted to get to that have started to come along, that overtime did come along. But as I was peeking over the fence I saw things are happening in the Python community and in the Ruby community that seem to be addressing the very problems that I was having: time to market, complexity, the ability to actually just, I don't know, handle a request that comes in from my server without a lot of obfuscation. So for me like a Web Forms developer, an ASP.NET Web Forms developer trying to do those simple websites, those simple things easily and quickly, and I thought that Jango was doing it really well, and I thought that Rails was doing it really well, and from my mind I just happen to fall in love with the community that was happening in Ruby and how the people were actually gravitating towards the technology and the way they work really impressed me so that was naturally where I gravitated to.

Richard Campbell: Because Ruby is not a young language. It's been around for quite a while and yet a few years ago it suddenly became rather hip.

Leon Gersing: Yeah. I think you could trace its hipness directly to Rails. I think that's the easiest thing. Ruby was around in 19..., I think '92 was when it was released. Matsumoto created it so if you have a term like MRI, that's Matsumoto Ruby Interpreter, that's the C-based interpreter that is what Ruby really is when we talk about Ruby typically at least. But yeah, it was relatively slow for the time but that wasn't its main focus. One of the things I've heard Matsumoto say is that one of the key building box, one of the foundations of Ruby is this concept of beauty in code, the true marriage of form and function. So it's not necessary that it has to be the most highly proficient language on the planet. There are plenty of languages that do that, but it's the one that can most elegantly speak the language of the domain. It's the one that really bridges the language gap between the business and the developer, and the developer and ultimately the machine. That's something that really appealed to me and I think the community really picks up on that. In fact, one community member who is kind of up and vanished in the last week, Why or Why the Lucky Stiff, he wrote one of this great Ruby manuals called Why's Pointing a Guy to Ruby. That was so bizarre, it was out in less field, they had nothing but cartoon boxes and crazy idioms in it but it taught Ruby in such a beautiful and elegant and mad cap way that I was hooked, I was fascinated with not only the language but the story.

Carl Franklin: Now, when we talk about Ruby, you say Rails is sort of the reason why people fell in love with it. Rails is a development environment for web applications that sort of has a lot of built-in scaffolding -- and for those who don't know what that is, it's just sort of a support code that is already sort of there. Isn't that true?

Leon Gersing: Yeah. I mean, that's a decent way of putting it. I mean, Rails is essentially a web framework, that's it. It's a framework for building web application.

Carl Franklin: So it's not a dev environment.

Leon Gersing: It's not a dev environment. Most people work in Emacs because they're more text-based, or whatever you want to work in that's your IDE, that's whatever you want. But Rails itself is essentially a -- I've heard it described as a Domain Specific Language or DSL for writing web applications. Everything you write for your application, you're writing in Ruby and it's just the Ruby dialect of building a web app. That's really it.

Carl Franklin: It's really a language custom build for your web applications.

Leon Gersing: Yes, absolutely. So if you're setting up your data layer and you need these various

schemas to be inputted at various times, there's a class that deals with what we call migration which actually defines schemas but does it in Ruby. So if you're defining something in Ruby and the minute it goes up you also want to add a bunch of data, you can add that data in Ruby configuration. Instead of having a mountain for pointy stabby XML files, we'll simply do our configuration in a compiled language that can be tested in Ruby. It's just that. We treat data as code, and code as data in Ruby and we might as well live by that and that's how we've done it, and I think that that makes it very easy, it makes a very low barrier of entry for developers who are trying to just get the job done and in that world it's getting the job of writing a web app done.

Richard Campbell: And just trying to get pass this whole XML as pointy and stabby thing.

Leon Gersing: Yeah. Well, someone argued that XML was never really meant to be human readable, that it's meant to be parsed by a machine, that's its job and it does it very well. You know, I'm not going to vilify XML. XML certainly has a place, but I think it has taken on this kind of role of the dynamic configuration bit in Java and in .NET and I don't necessarily know that most of those things are -- I don't know that they are appropriate for the level of abstraction you're trying to gain. I think they usually end up being this runtime hacks where we put some stuff in, we read up the XML, we parse it, put it into some other domain object which is really just a direct reflection of the serialized version of it which is the XML, and then from there we do some other stuff to it. Well, we just cut up that serialization point, just put it all on Ruby. Done.

Carl Franklin: This portion of .NET Rocks! is brought to you by Telerik without whose support this show surely would not exist. You know, summer is peaking and our friends at Telerik are working full steam. They've just released the Q2 volume of the Telerik premium collection for .NET which is their biggest release yet. Packed with new things, it will surely excite anyone who has anything to do with .NET development. Let's start with Silverlight in the introduction of the first commercial 3D chart on the market. It is developed as true vector 3D which guarantees swift performance and rich capabilities like rotation, animations, etc. Another major announcement is the Telerik Silverlight Scheduler which is packed with tons of features even in the first version. Telerik's flagship, RadControls for ASP.NET AJAX, grows not only with four new controls but also with new productivity tools. Take the new Visual Style Builder, an online application that allows you to visually modify skins or design new ones with point and click. If that's not enough, they've added a completely new product, a free testing framework powered by ArtOfTest for automating AJAX and

Silverlight Rich Internet applications. Since I'm short on time here, I can't enumerate all the new features and enhancements in Telerik Reporting, OpenAccess ORM, and their Windows Forms products so I'll leave it for you to check them out at telerik.com, and don't forget to say thank you for supporting .NET Rocks!

So for those who have never used it, I'm trying to -- you know, when you're naming off all those things, well, I just do it in Ruby. I mean, I could say the same thing about C# from not having seen too much Ruby.

Leon Gersing: Yeah, sure. Most things -- I mean, they are application level languages. They're object-oriented programming language, C# and Ruby both serve that purpose. They were strongly typed. The difference is coming at dynamism, if I can say that word. It comes in the fact that while they're both strongly-typed, Ruby is dynamically-typed whereas C# is statically-typed which allows us to make a lot more declarations at runtime. We have differences of opinions on those two languages on what Inheritance means, what is the best way to do inheritance. So in C# you've got single level Inheritance, a single class Inheritance, and then you can create basically these interfaces that are ways to decide, okay, this particular class will respond to this contract. It's very static. It's very this is what it is, it's not really going to change. If it changes, it's a recompilation and we've got to deploy more code and that's it, and that works for some situations. In some situations you're going to need the speed of those compiler pre-optimizations and that static language is going to provide that, but when you don't, when you can mitigate that in some other way you can assemble your classes at runtime. You can assemble your object graph at runtime as needed. In Ruby we have single objects inheritance, but we also have the idea of a mix in. So we have modules that can be mixed in at any time that allows to extend, reduct, change, basically change code as it leave and that really helps us get rid of some of the klugey mechanisms that we've seen that have kind of come up in the Java and in the C# world like Dependency Injection. Dependency Injection is something that exists because those are statically-typed languages that can't necessarily do the evaluation at runtime in an elegant way so we have to load them up from configuration and figure out what we're doing, set types statically, and continue. Whereas in Ruby we could just, you know, add it at runtime and do whatever we want. We can make any class anything we want it to be.

Carl Franklin: So how does its dynamism make it a class that's suited exactly for web development?

Leon Gersing: I don't know if it's just suited specifically to web development, but a lot of it would have to be -- I mean, yeah, I don't even think it's

limited to just web development to be honest with you. I think anything where runtime expectations are -- even Cabal Time, all right, so design time or runtime expectations are unknown, various bits might be not known. It makes it a lot easier to define the meta of the domain and handle that. So let me give you an example. So in the Active Record, which is the ORM for Rails, there's the notion of the dynamic finder. So in most repository patterns, there's this idea of Model.find and then you can pass in whatever criteria you need to find a particular output. In C#, it's going with repository in the specification pattern. So we use the Lambda to determine exactly what object we want to, or exactly what data we want to retrieve from our object store. So if we get rid of that and we go with the dynamic finder, then we can have a very clean API that reads more like human language and reads more like the domain we're trying to specify. So there's the base one which is its spine so that's the one that's defined. So there's Object.find and you pass in an ID or a key, the primary key, and it goes and fetches the object. But we don't necessarily know what every object's properties are going to be but we may want to search by them later. So we may have a blog post and so the blog post can have a title, so we may also want to find by title. It may have a slug so the concatenated URL, so we might also want a method that is find by slug. Now, it would be very tedious to do this in a statically-typed language. You have to define each method. You're probably going to be reusing the same method over and over again. Your code would get littered and peppered with all of these finders that you want. Well, since Ruby is dynamic and it has message passing, and we can talk about that in a minute, you can implement one method called Method Missing which will catch any method that that particular object or class will not respond to, and at that point we can have some smarts around it. So we find, and if you pass in find_by_X or whatever X happens to be, we can be smart about it and say, oh, well, X represents the slug so let's build the actual SQL statement to find the slug for this name and let's add this method that you've just asked for directly to our class so that any other instances of this particular class can also ask for find by slug and get it. So we'll basically emit a new method to that class, on that class exactly, not a reflected...

Carl Franklin: Right.

Leon Gersing: Mock proxy class, the actual class. We're going to change that to actually fit our needs and each subsequent call to it will be fast because it's already a member. So that's the kind of dynamism that really makes something like Rails shine, keeps your code concise, keeps it very simple.

Carl Franklin: So I guess the point is that the dynamism works really well in ASP.NET or in any kind of development.

Leon Gersing: Absolutely. I mean, these concepts are universal. I mean, we want to write less code. We want to code the ideas of our domain. We want to make sure that the concepts are mapped. We don't want to be stuck writing the same boilerplate code that could literally be generated for us either by the language itself or by something else to kind of mitigate the crap that nobody wants to do.

Richard Campbell: Does it make it challenging to debug an app when it's this dynamic?

Leon Gersing: I think you can and I think that certainly new Ruby-ist tends to be overprotective in their code. Sometimes though if you're coming from a statically-typed language, what I'll see is I'll see a lot of type checking so basically they still type system in Ruby but people add it.

Richard Campbell: It's create their own, you know.

Leon Gersing: So is it type this, is it type that. That actually goes against some of that dynamism. You're actually kind of taking it away if you're always type-checking. We have something in Ruby called Duck Typing which allows us to basically just ask does a particular instance or object that's been sent to me, does it respond to a method? If it doesn't respond to a method, then I can simply just move on. They don't have to invoke it on there or anything, but if it does I don't care where it came from. This is Duck Typing so it looks like a duck, acts like duck, quack like a duck, it's a duck. So if you send me a dog and I say can you quack and the dog says yeah, I quack, then I can just ask it to quack and we're fine. We're happy.

Richard Campbell: I don't really care whether it's a duck or not.

Leon Gersing: I don't care. It doesn't matter.

Carl Franklin: So you don't need to query to see if it has an interface implemented. You just wanted to know if it has a quack method.

Leon Gersing: Exactly, exactly.

Richard Campbell: I guess that's the clever bit there. It's just its ability to -- well, the other aspect of this I guess is everything has to be an object then because anything can have a method. We're talking about a variable here.

Carl Franklin: Yeah.

Leon Gersing: Yeah. Well, anything and in fact everything in Ruby is in fact an object.

Richard Campbell: Right.

Leon Gersing: The class itself that defines other objects or instances of that class, that class is an object and in fact you can actually send methods to the class of an object instance if you want to. It's a very, very complex and interesting system and it all basically revolves around message passing.

Carl Franklin: So I totally understand your love affair with Ruby. Do people argue that Ruby is great for simple websites, but for really complex things where there is a lot of stuff going on Ruby might not be the right tool for the job? Like when is Ruby not the right tool for the job?

Leon Gersing: I've yet to come across the scenario in my life where Ruby was not appropriate. Now, there are times when Ruby can be supplemented with another language that would be more appropriate.

Richard Campbell: Interesting.

Leon Gersing: So if I needed things like, you know, I was going to do a messaging system. I may write the whole thing in Rails, but then when we got to the actual messaging part that maybe a little too slow and Ruby doesn't deal with concurrency as well as it could but I can flip out to something like OCaml, or Erlang, or something else that's really good at just that one part and I can write 20 lines of Erlang to do that messaging bit and still stay in Ruby for everything else.

Carl Franklin: Right.

Richard Campbell: I was thinking that something really mathematically intensive might be better served in a language that's really heavily math oriented.

Leon Gersing: Absolutely, absolutely and that would be the first thing I would say. You know what? This isn't Ruby, let's ship that off. And Ruby is really good in dealing with the operating system no matter which one it is so it can just simply call out to whatever it does...

Carl Franklin: Yeah.

Leon Gersing: Get the return value and continue processing as it would, and hopefully I would think that people on .NET framework, once they have a full DLR version of IronRuby, I would hope they may even look to C# as that sub-language or even F# as that sub-language.

Carl Franklin: Are there times that you miss things in the .NET framework that you wish were there?

Leon Gersing: Well, that's a set-up. Here come all your emails, you can just forward these on to me. No, there's really not a whole lot. Well, it's nice to have a vendor to blame every now and then. That's nice. I don't deal with many vendors at the moment so if something goes wrong it's nice to be able to call up my evangelist or somebody and say, hey, can you give somebody a line to fix this? But on the other hand, the person who gets to fix it usually is me because I can.

Richard Campbell: Right.

Leon Gersing: But now I would say there are certain things that are nice about the .NET framework. Maybe, I don't know. That IDE is really nice.

Carl Franklin: Yeah.

Leon Gersing: There has been a lot of time making Visual Studio really nice and I miss ReSharper every now and then.

Carl Franklin: I guess you're not – you know, in a web application, you're not necessarily using a lot of plumbing stuff, that IO and the diagnostics, and all that stuff.

Leon Gersing: Exactly. You know, not really.

Richard Campbell: So what about IronRuby? Because this is going to be a version that really lives in Studio.

Leon Gersing: Yeah. I'll tell you what, IronRuby is -- I mean, if you're at all interested in Ruby and you're on that platform, on the .NET platform, please, please go look at it and maybe contribute to it. The implementation is done in C# so it's a pretty one, for one. First and foremost, they're basing the entire thing off the Rubinius Test Suite. So Rubinius is basically a project in the Ruby community that looks to basically write Ruby in Ruby so it's entirely a complete language. So to do that, because there's no real spec, they essentially unit test it all of Ruby, the language.

Richard Campbell: Right.

Leon Gersing: At least the MRI version of it. So from my Rubinius' perspective, they're actually going against that test suite to see if Ruby is feature complete with the C version of Ruby which I think is amazing. So Microsoft has the tendency and have done in the past where they see a great idea that's happening outside of their domain and instead of learning and integrating with that community and becoming a full member of it, they co-op it, they take it for themselves and change it into something. I think

we could look at something like MSTest as a clear example of where that's a failure. So on the IronRuby side, yes, so they're going against Rubinius, it's going to be a feature, a feature compatible with C version of Ruby which means you cannot take advantage of all the libraries that exist but as Rails runs on it, it just runs really slow. But yes, once it's there I think that the first and foremost is I think people should look at it if they're trying to test things on the framework that are private or sealed. So there's a lot of hobble that was going around about, hey, we can't mock the HTTP handler or something, I can't remember, all the HTP context.

Carl Franklin: Because they're sealed?

Leon Gersing: Yeah, they're private and sealed so you can't inherit from them, and mocking them is an Herculean task at best and you can't really get in there and replace it the way you want.

Carl Franklin: Can I ask you a rhetorical question real quick?

Leon Gersing: You ahead.

Carl Franklin: Why would anybody seal a class?

Leon Gersing: That's a really great question and you're going to have to get somebody who likes that keyword to give you an answer. I think access modifiers are really the true evil.

Carl Franklin: Yeah.

Leon Gersing: The only reason I would say mark a class private is if you really don't want to expose it in the API.

Carl Franklin: For legal reasons.

Leon Gersing: As a courtesy to your developers or people who might use it, but otherwise keep that stuff public.

Carl Franklin: Yeah, really.

Richard Campbell: It almost strikes me as people are embarrassed by their class so they're hiding it.

Leon Gersing: I hope that's not the case. They need to write up their unit test if they're that afraid of it.

Richard Campbell: Yeah. I just can't think of a better reason.

Carl Franklin: I don't know.

Leon Gersing: I think some people do it in line with obfuscators.

Carl Franklin: Right.

Leon Gersing: When something is private and it's sealed, many top line obfuscators will turn it into this weird character thing so you can't even recognize the method signature.

Carl Franklin: Security fears up security.

Leon Gersing: That's horrible. That essentially is saying I don't trust you to use this.

Carl Franklin: Yeah.

Leon Gersing: It says I know better than any developer who's going to touch this code, and I just can't get behind any language that says that to me.

Richard Campbell: Yeah. I just see it as a mistake.

Carl Franklin: This portion of .NET Rocks! is brought to you by our good friends at Red Gate, makers of ANTS Memory Profiler. You know what a Performance Profiler does. You run it and it tells you where the bottlenecks are in your code. You can profile any .NET application including ASP.NET web apps. So if you're a .NET developer of any kind and you want to find out where your code is choking, go to shrinkster.com/19op, that's one, nine, the letter O, P as in Paul, and check out Red Gate's ANTS Performance Profiler. You'd be glad you did.

Anyway, you're making a point when I rudely interrupted you.

Leon Gersing: Oh yeah. So IronRuby in Rails, yes.

Carl Franklin: Right.

Leon Gersing: So they were using this conference driven development style for a while so, you know, a Rails conflict come up and they go, okay, let's get IronRuby to run Rails, and they did and it was slow, but I would say a great entry point is testing. Now, I've heard rumblings and hopefully somebody on the IronRuby team will write in a great letter saying that I'm absolutely right and they're not looking to change this. But from a testing point of view, if you want to mock out certain classes on the CLR that are private and sealed and you can't get to them any other way, you can actually mock that in Ruby and IronRuby as a full mock and you won't be using a generated proxy version of that class. You'll actually just be re-implementing the various members on that class that you want mock. It gets you everything

you've ever wanted in testing in the unit testing framework without having to leave say C# to get it done. You just have to pop out, create a low IronRuby class that does the stubbing for you or the mocking, and then call it from C# and you should be all good to go.

Carl Franklin: Huh.

Leon Gersing: In fact on a -- MX Lab did a new project called Gestalt in which it was basically trying to give you any DLR language in the context of the browser. So the browser, when you say Script Tag you usually talk about JavaScript, that's what the browser talks. What Gestalt is trying to do is give you DLRs so you get Python and Ruby in their Iron format. So I actually did some of the demos for that site and one thing they wanted to do is Falling Snow, one of these nice, cute Falling Snow effect and because of this version of IronRuby I couldn't get the Inheritance to work, it was saying this user control can't be inherited for some reason, but that was just a small bug and it was a much older IronRuby version. So I'm sitting around, I say, well, what do I really want to do. Oh, all I want is an Ellipse that can move, so I wanted one method and maybe four properties, a Move method and a couple of properties to track its motion. So all I did was instead of inheriting from Ellipse, which is private and sealed by the way, I simply opened the Ellipse in the same namespace and just added my methods on and then I was able to just move as many Ellipse as I wanted to. Happy as a clam, I didn't write a whole bunch of crap code that was a series of crazy Inheritance, I didn't have to any of that, I didn't have to implement a user control at all. All I had to do is open Ellipse, put on what I wanted and let it go.

Carl Franklin: Hmm.

Richard Campbell: Because part of this is just coming up with an elegant solution of the problem too.

Leon Gersing: I think that's what all of this is. I think when you ask the question at the beginning, is software getting complex, I would say yes, it's getting complex especially on the .NET side because the abstractions are further and further away from your power center. The power center for a developer, the thing where the chi is formed is your language, it's how you speak to the computer.

Richard Campbell: Right.

Leon Gersing: And if you just keep getting away and away and away from that part of it, you're eventually going to get to a part where it doesn't look anything like it should, that you're completely disconnected from the entire pipeline of what's going on, how can I talk to the computer in a way that it's

going to understand and my user is going to understand. I think ASP.NET itself, the Web Forms version of it is a great example of this. This is a problem in which it was trying to solve it in a way that the abstraction was broken.

Richard Campbell: Yeah.

Leon Gersing: It just leaks all over the place. So they were taking this desktop MIM and trying to make it a web MIM. It doesn't work. It didn't work.

Richard Campbell: It seems like ASP.NET is not moving away from that. MVC seems to be a much better approach to building something closer to the mettle of building web apps.

Leon Gersing: Absolutely. I think it's a wonderful addition to the family of products. I wish they would have realized this a lot sooner but that's the way you turn a big ship, it's very slowly.

Richard Campbell: Even up to just a few months ago, they were talking about "Don't worry, MVC is going to have relatively low adaption. It would just want, for the EdgeCase, that we wanted a product for them and it seems like a real storm around it now. People are really going nuts for MVC.

Leon Gersing: Well, of course. I think whenever you get trap into looking at the solutions for the platform you're on, and it happens in any platform like I get mine when I think about Ruby. I'm guilty of it just because that's where I'm at.

Richard Campbell: Yeah, you've got the Ruby hammer and you're looking for nails.

Leon Gersing: Yes, exactly but I will be fair. There are times where I will bounce out of it if I need to.

Richard Campbell: Yeah.

Leon Gersing: But yeah. I mean, yeah. But in Ruby I always see this nail and I ask if whatever I'm holding will respond to hit that nail.

Richard Campbell: Right.

Leon Gersing: And it usually ends up as true.

Carl Franklin: In Ruby, can you shell out and run managed code?

Leon Gersing: Yeah, sure.

Carl Franklin: So you can call assemblies from Ruby?



Leon Gersing is Having a Love Affair with Ruby!

September 17, 2009

Leon Gersing: Yeah.

Carl Franklin: I don't mean IronRuby.

Leon Gersing: No, no, no. Yeah, you can shell out and do whatever you like.

Carl Franklin: Hmm.

Leon Gersing: You can go nuts.

Carl Franklin: You can call like regular assemblies. You don't have to write them as console apps and you standard in and standard out, right.

Leon Gersing: You could do either one, but yeah you can.

Richard Campbell: I get the sense like it's very inclusionary, that you could go about this the way you want to.

Leon Gersing: That's why I like it, that's why it works for me, that's why I prefer to look at Ruby the language for a problem than look for a tool to solve it. I mean, things like Windows Workflow Foundation to me seems large and verbose and XML driven, and if you're not using XML, you're using a mountain of this crazy apps that were written to hopefully talk about the Workflow in a way that maybe maps the way that you wanted to do, that's all crap to me. I mean, I can write a state machine Workflow using the access state machine gen in 10 lines of code and be done with it and it's fully unit tested.

Richard Campbell: Yeah, there's just a better way to go about that. Going back to this complexity issue, I think what we're really talking about, you know, the big difference between there are 71 different MVC implementations for PHP and the situation we have at Microsoft is that this is not about there's lots of choices to do the same thing. This is there's a whole bunch of ways of doing different things, and Workflow is a great example of that of do I need this? Am I saving effort here if I go this way?

Leon Gersing: Yeah and that's a tough question to answer, and when I was in that space a lot of times we would always end up on let's try the tools first and it worked great for the 80% of what we needed and the minute we got to that 20 I ended up having to back it out and put in all custom plumbing.

Richard Campbell: Yeah and it's a lot of time.

Leon Gersing: Or some clugy hybrid and it became completely unmaintainable and the fact that I couldn't accurately and actively unit test, get 100% unit test coverage over all the pieces I was writing because I might have been working with something

say like SharePoint, it made it impossible for me to even feel confident about the code I was shipping. Those things to me are just black, black spots in that framework that I really wish would get alleviated, and I think something like IronPython, IronRuby, they maybe the key for at least me and the way I look at software helping me be happy around that platform.

Richard Campbell: What are usually the differences between Python and Ruby in this? Because they both seem to be lumped together in some degrees.

Leon Gersing: I mean, they share idioms than say -- I mean, they're both dynamic languages.

Richard Campbell: Right.

Leon Gersing: That's really is. But they have similar approaches to certain problems and they have different approaches and different problems. I don't want to detract from Python by just saying Ruby and I think that's why I tend to try and talk about them together.

Richard Campbell: Uh-hmm.

Carl Franklin: Have you talk to John Lam about IronRuby at all?

Leon Gersing: Unfortunately no, I have not.

Carl Franklin: Is there anything you want to tell him?

Leon Gersing: Get more resources on the titanium.

Carl Franklin: Yeah.

Leon Gersing: I would like to see the DLR treats it with the same respect as the next product that they're trying to sell. Unfortunately I don't think they think the market is there for it. So like the Oslo team, I don't know how many people are on that team, but it seems like it's gotten a lot more attention and hype than say IronPython which just is well passage to your old version birthday. It's 2.0 version.

Carl Franklin: Right.

Leon Gersing: So there are things that you can do and you can sell leveraging an IronPython if you want to, that you don't have to wait for a product to solve that problem. You can actually solve that problem at the language level and be well on your merry way in something that's flexible and maintainable.



Leon Gersing is Having a Love Affair with Ruby! September 17, 2009

Carl Franklin: So what are some of your other favorite technologies, Leon? I know JSON is a big one for you, right?

Leon Gersing: Well, JavaScript in general is huge for me. I've always believed in the power of that particular language. I'm interested to see how people are starting to use it in their solutions. One particular piece of technology that I'm speaking on soon is called Titanium by Appcelerator and they're actually building out native mobile solutions, a native iPhone, native Android using basically a JavaScript bridge so you write the app in JavaScript and it compiles down to native Objective C or whatever it turns into on the CoCo side and then...

Richard Campbell: No kidding.

Leon Gersing: Yeah and oh, it's amazingly doing a great job. The entire stack is open source so if they're going to a direction you don't like, fork it and go nuts. You can go along and get Hub but it's a great idea.

Richard Campbell: Isn't it interesting that they've got iPhone and Android listed here, but no Win Mobile?

Leon Gersing: Now, there's a big push from the community and including myself to see if they can get that Win Mobile support as well. There are other teams that are doing it too like Rho mobile. They're doing multi-platforms so they support Palm Pre and all that, but it's not native. It boils down to like a web, they use the web browser and all that.

Richard Campbell: We are in an interesting place now where we've never had a better mobile platform development out there than right now. It's just incredible and people are trying to make great tools for this, and the one group I don't see doing anything here is Microsoft.

Leon Gersing: Yeah. From the OS perspective or for the tools to build for it?

Richard Campbell: From the tooling perspective like yeah, right, the Mobile OS, let's not even talk about that.

Leon Gersing: Yeah.

Richard Campbell: But a problem of course here I guess is that they go hand-in-hand because the Windows Mobile platform hasn't been going well. The Windows Mobile Development Tools haven't been going well.

Carl Franklin: Well, and there are some nascent things going on there too that I think you're

going to hear about soon and I don't know what that is, nor can I speak to it, but they've sort of been hibernating a little bit.

Leon Gersing: And also I think they have all the pieces to make it work really well. If they can get a great mobile version of the CLR, I know they have the compact edition but it's really lacking in certain features, but if they can get a great version of that and maybe be able to run something like Silverlight 3.0 on it., then I think if you can have people targeting web apps and what-not, that will be a great first step for like a native implementation but that's a possibility, we'll see.

Richard Campbell: Yeah, I think you're right. The Silverlight could save the Windows Mobile platform.

Carl Franklin: Oh, it definitely could.

Leon Gersing: I mean, it's great, it's compact, it's simple, people are used to that Silverlight version of the CLR. I think if you put that on a mobile phone, you can really deliver any kind of user experience that you could want to, and then you have language choices, you have platform, that great. I think that would be wonderful.

Carl Franklin: I do too. I think they ought to stop making -- I don't know, stop leaving the look and feel up to the phone vendors. I mean, just pick something and go with it so we don't have to drive through 50,000 menus with the Stylus to get to place a phone call or something.

Leon Gersing: Absolutely.

Richard Campbell: Yeah, funny place. Is there room for Ruby in the mobile development?

Leon Gersing: I don't know. That's a good question. I've heard on the JRuby side that they're targeting getting a Ruby emulator going for mobile, for Android. But yeah, I don't really know, and I think MacRuby which is the Mac version, it's like a CoCo bridge included in it., it's looking to get on to the iPhone as well, but I mean I think it's one of those things that we'll see. It's a pretty resource constraint device.

Richard Campbell: Yeah.

Leon Gersing: And Ruby can, yes of course, be a little slow and that depends on the device. Maybe the iPhone can handle it, maybe an Android device can handle it, I don't know.

Richard Campbell: I also get to sense the constraints are going away. You know, ARM

announced this year that next year they'll be releasing multi-core processors for cell phones.

Leon Gersing: Wow.

Richard Campbell: I feel like in the next couple of years, there's going to be as much horsepower in our PDA, it's the reason it's a typical desktop machine.

Leon Gersing: Then we have the same arguments that we had for Ruby 10 years ago.

Richard Campbell: Right.

Leon Gersing: That Ruby wasn't ready because it wasn't fast enough. Well, the machines caught wind of it, we sped up, then it happened.

Richard Campbell: You're right.

Leon Gersing: I think if Ruby doesn't catch on for the mobile platform, it certainly might be whatever Ruby is next predecessor is.

Carl Franklin: I think Ayende started working on the portable nuclear reactor, the power of these devices because...

Leon Gersing: Of using ColdFusion.

Carl Franklin: If you can put it on your hip and you can just have a little reactor, maybe some insulation so you don't get too hot.

Leon Gersing: You can warm the coffee that way.

Carl Franklin: Warm your coffee.

Leon Gersing: You put it on top.

Carl Franklin: Just a little bit of plutonium is all you need, just a little bit.

Richard Campbell: What do you mean I can't get on the airplane? What?

Carl Franklin: What could go wrong?

[Laughter]

Leon Gersing: That would be great.

Carl Franklin: Well, you've sufficiently kicked our ass for an hour and that's great, man.

Leon Gersing: Well, anytime you guys needed me and want me to share, you can call me up.

Carl Franklin: You bet.

Richard Campbell: So what are you working on next there, Leon? Where are we going to see you?

Leon Gersing: Oh, let's see. Well, this week is probably out because it won't be recorded but I will be at the Ruby Hoedown, and then coming up is the Cincinnati Agile Roundtable talking about Enterprise level integration testing with Cucumber.

Carl Franklin: Cucumber.

Leon Gersing: Yes.

Carl Franklin: What's Cucumber?

Leon Gersing: I'd probably be doing a few more of those talks hopefully until the end of the year, and I think I'm slated to do some pre-compiler stuff at CodeMash this year on TDD.

Carl Franklin: You can't just drop a word like Cucumber on our laps and not define it. What is that?

Leon Gersing: Oh, yes, sorry. Cucumber is a framework in Ruby that does basically Behavior Driven Development.

Carl Franklin: Okay.

Leon Gersing: It's just an apps and text style. It's a great way to leverage your business domain, those business experts, and write out their acceptance criteria in a language that they provide that you can then execute against real Ruby code. Actually you can execute it against anything. You can use Cucumber with C# and Java, all of it.

Carl Franklin: Awesome.

Leon Gersing: I think there's info at huge.info.

Carl Franklin: Thank you, Leon.

Leon Gersing: Thank you guys, I really appreciate it.

Carl Franklin: It has been great. Come back again. And we'll see you next time on .NET Rocks!
[Music]

Carl Franklin: .NET Rocks! is recorded and produced by PWOP Productions, providing professional audio, audio mastering, video, post production, and podcasting services, online at www.pwop.com. .NET Rocks! is a production of Franklins.NET, training developers to work smarter and offering custom onsite classes in Microsoft development technology with expert developers, online at www.franklins.net. For more .NET Rocks!



Leon Gersing is Having a Love Affair with Ruby!
September 17, 2009

episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.