.NET Rocks!
The Internet Audio Talk Show for .NET Developers
With Carl Franklin and Richard Campbell
msdn
http://www.dotnetrocks.com

HTTP://www.dotnetrocks.com

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!

Text Transcript of Show #683
(Transcription services provided by PWOP Productions)

Pwop! Productions

Mark Rendle Has Some Simple.Data

July 26, 2011

Our Sponsors

RadControls FOR ASP.NET telerik

http://www.telerik.com

Lawrence Ryan: .NET Rocks! episode #683, with guest Mark Rendle, recorded live Wednesday, June 29, 2011.

[Music]

Lawrence Ryan: This episode is brought to you by Telerik and by Franklins.Net - Training Developers to Work Smarter and now offering video training on Silverlight 4.0 with Billy Hollis and SharePoint 2010 with Sahil Malik, order online now at franklins.net, and now here are Carl and Richard.

Carl Franklin: Thank you very much. Welcome back to .NET Rocks! It's .NET for an hour with Carl and Richard and special guest. Hey, man. What's up?

Richard Campbell: Not much. How about you?

Carl Franklin: Oh, you know, I don't really have any banter today.

Richard Campbell: You're banterless.

Carl Franklin: I'm bantered out so I guess we'll just jump into Better Know a Framework.

Richard Campbell: Awesome.

[Music]

Richard Campbell: Well, what have you got?

Carl Franklin: Yes, so I like to go back and forth between classes, and projects, and resources. So I was looking at CodePlex in the most popular downloads and I came across this interesting project from Microsoft Spain. It's a domain-oriented N-layered .NET 4.0 sample application.

Richard Campbell: Interesting.

Carl Franklin: And it's at -- oh, let me see if I can read the small URL here, and it's at microsoftnlayerapp.codeplex.com, all in one word, microsoftnlayerapp.codeplex.com. The project scope says it's a sample implementation of most used patterns in Domain Oriented Architectures based on simple scenarios easy to understand such as customers, orders, bank transfers, etc. It shows Isolated scenarios useful from a patterns implementation point of view, but not necessarily as a whole business application. The architecture is decoupled and relatively complex so it would allow you to grow your system. I like this because everybody needs those sample apps to just sort of see what the end result should look like.

Richard Campbell: Yeah, what it should feel like, all those different pieces.

Carl Franklin: That's right.

Richard Campbell: That's why I love this, domain layer, app layer, data persistence layer, presentation layer, distributed services layer. This is the way it should be.

Carl Franklin: Yeah and most samples are either too stupid or they challenge your -- they're too complex. This one looks pretty good and the reviews are great. "Great work! Trendy technologies in a professional architecture ready for the real world." Good stuff. So Richard, who's talking to us?

Richard Campbell: I grabbed a comment off the website from Show 664, which if you recall is the Radio Astronomy show.

Carl Franklin: Oh, that was fun.

Richard Campbell: One of my favorites. We had such a good time doing that show and the guys were amazing. We got a comment from Shawn De Wet and he says, "Hey guys. As a South African I was listening somewhat more intently at this one since our Kalahari desert is one of the candidate sites for the Square Kilometre Array and having Australia as our competition makes it a bit more 'personal' too since the rugby history between the two countries has ensured that, in the event of the Square Kilometre Array going to Australia, we will certainly feel that we lost a competition and we HATE losing to Australia. Anyway, something you said during this discussion made me feel this loss coming on in a big way. You mentioned the enormous power requirements of the project to the extent that "a small nuclear plant" may suffice. You see, we had nasty power outages in 2008 and South Africa has been on somewhat of the "power rationing" mentality, and the public has been led to believe we are on the brink of a power crisis. Old mothballed power stations have been re-commissioned and two new HUGE power stations are under construction, but their effect won't be felt until late in the decade. Sadly, this has to do with our history and how the power infrastructure that used to serve about less than 10 million people is now expected to serve about 40 million people bringing electricity to every home and they realized it a bit late and the existing infrastructure would just not handle it, hence, the current situation. So, I expect us to lose to Australia, but at least it's not the rugby field. I think a .NET Rocks mug might help me drown my sorrows."

Carl Franklin: Yeah, and also maybe think of a way we can change the world power wise. You know, there are a lot of smart people out there that listen to our show. Here's a challenge. Solve the power problem.

Richard Campbell: Well, and you know what's interesting is South Africa was also one of the leading developers in pebble bed nuclear technology.

Carl Franklin: Which is?

Richard Campbell: Pebble bed is a totally different way of using nuclear technology that is a much lower risk. As the heat goes up, it naturally reduces its power so there's no way to have a meltdown. It also runs in what's called a medium cycle. It's a lot lower temperature in general. So it can use less pure uranium. You don't have to do as much refining and so forth. In fact, if it works the way it was supposed to work, the current spent fuel rods that are sitting in on nuclear plants all over the world would be fuel for a pebble bed reactor.

Carl Franklin: Guys, what I just told you about solving the power problem, never mind. Richard just did it.

Richard Campbell: The problem is it's a very difficult technology to develop and the South Africans are pretty much the only ones working on it since President Clinton shut down the Hanford research facility in Washington which really is frustrating.

Carl Franklin: Oh, darn.

Richard Campbell: Because I felt like we were right on the cusp of a revelation in power generation in that space.

Carl Franklin: Wow.

Richard Campbell: It's just not being worked on anymore because it's expensive. We're talking billions of dollars in research, but the outcome could be stunningly good. It's one of the reasons I read this comment is that it's near and dear to my heart that we come out with better solutions to power.

Carl Franklin: Yeah.

Richard Campbell: We've got these great technologies just so close. We just have to pour some energy into it. Anyway, Shawn, I'm happy to drown your sorrows. There's a mug on its way to South Africa. If you've got questions, comments, ideas, even if they're not necessarily technology, I'll happily talk electricity all day, send us an email at dotnetrocks@franklins.net or write a comment on any show at the .NET Rocks! website.

Carl Franklin: Before we send you that mug, we're going to break it in with a little Woodford Reserve bourbon.

Richard Campbell: Of course.

Carl Franklin: We got to wash it out after all.

Richard Campbell: We've got to do something.

Carl Franklin: We don't want to send you any airborne or food borne diseases. Richard, it's a really interesting show today because our guest is none other than Mark Rendle. Mark is the man behind Simple.Data. He is currently employed as principal software architect at Dot Net Solutions Ltd where he creates all manner of software and services on the Microsoft stack including ASP.NET MVC, Windows Azure, WPF and Silverlight. He has been a Windows Azure Development MVP since April, 2011. In his spare time, Mark works on the Simple.Data not-an-ORM project and builds developer-centric tools for mobile devices including the award-winning Pocket C# for Windows Phone 7. Welcome, Mark.

Mark Rendle: Hi.

Carl Franklin: So you do all this work. You're inventing all sorts of great software, you're married, you have a baby. When do you sleep?

Mark Rendle: Sleep is for the weak.

Carl Franklin: It's a cling-on phrase, isn't it?

Mark Rendle: No. I go until I drop. If I say I'm going to sleep now, then my brain just goes. So I just keep going and doing whatever it is I feel like doing until I actually pass out and then my wife comes down and find me at 4:00 in the morning on the sofa. I just hope that I manage a good kind of commit in the pushing before that happens.

Carl Franklin: I remember those days fondly. It sounds like you're recalling a moment from my past. Well, let's talk about Simple.Data. You say fervently that it's like an ORM without O, R, or M. What do you mean by that?

Mark Rendle: Okay. So it's dynamic-based and so there aren't really any objects or not in the sense that ORMs have objects, and it's also Relational Database but it also has the ability to talk to any other data source you can think of. So someone has written that to get it to MangoDB so you can fire the relational app. The mapping thing isn't really there because what it actually does is it pull things after the database and present them as dynamic, and then most dynamic objects do have the ability to cast themselves to statically-typed objects but you don't have to so there's not necessarily any mapping going on either.

Carl Franklin: So essentially it's an API that returns these blobs that are just data structures that are dynamic, made up on the fly type list I imagine.

Mark Rendle: Yeah.

Carl Franklin: Since you use them directly, you don't have to map them into entities.

Mark Rendle: Yeah, that's right.

Carl Franklin: So I also imagine that you would not be using this with Entity Framework or something like that. You'd use it in lieu of any other kind of...

Mark Rendle: Yeah. You use it instead of Entity Framework or NHibernate over the Code First or Fluent versions of those.

Carl Franklin: Right and are these blobs that come back? Are they bindable blobs?

Mark Rendle: You can use them as the model in an ASP.NET MVC page and you can bind directly to them from XAML in WPF 2011. I haven't actually got a build working in Silverlight yet, but that is something that I want to do in a while.

Carl Franklin: Right. After you sleep.

Mark Rendle: After I sleep for about a month, yeah.

Carl Franklin: So what's the downside? I mean obviously are there any trade-offs? Are there things that you're trading off for this kind of dynamic flexibility?

Mark Rendle: Yeah. The big trade-off is that because you don't generate a repository, you don't generate like an entity context, and so you don't get IntelliSense. That's the big trade-off if you leverage a C# or VB.NET developer, it's that you say Database Open, give it a connection string and then that gives you back a dynamic object and then you say Database Stop and it just gives you the full basic object methods that all .NET objects have and that's it. There's no kind of prophecies on there and so you lose that element of discoverability.

Carl Franklin: If they're dynamic, could you not generate enumerations or something?

Mark Rendle: I've been looking at writing an IntelliSense plugin that will actually hook into the database at design time and see if I can inject IntelliSense into Visual Studio. If it knows you're looking at your customer's database, then it will show you properties for all those tables that are in there.

Carl Franklin: Is that your number one request? IntelliSense?

Mark Rendle: No, not really. My number one request, I've kind of dealt with them all actually.

Carl Franklin: Okay.

Richard Campbell: So it's perfect.

Carl Franklin: It's perfect.

Mark Rendle: People ask for providers or adapters for different data sources.

Carl Franklin: Yeah.

Mark Rendle: And since I've done, over the last few weeks I put in a full querying system which can do kind of grouping and explicit joints and also some of the stuff that you can eventually represent the entire body of a SQL select statements by using typical data kind of dynamic, so much the DSL or any, and that now has made most people happy.

Carl Franklin: Well, that sounds an awful like LINQ.

Mark Rendle: It is. It's very, very much, like it looks so much like LINQ but you haven't had to generate thousands of lines of XML into your project in order to be able to write LINQ looking statements and so it's a lot faster than LINQ to SQL and the hell of a lot faster than Entity Framework.

Carl Franklin: Did you think of just implementing iQueryables so that you could snuggle up to LINQ?

Mark Rendle: I did and I looked into it and David Fowler at Microsoft also looked into this when he was doing the WebMatrix Data Access component. The problem with iQueryable is the Expressions Count includes dynamic types.

Carl Franklin: Oh.

Mark Rendle: A Funk can take a dynamic parameter, but an expression of a Funk Count take a Funk that takes a dynamic parameter. So you can't do all that without getting in there and jumping around and ripping things apart. The queryable thing is out of the question at the moment and I'm not expecting Microsoft to particularly go back and pile more stuff into dynamic because I think those guys have a lot to make. They kind of get a feature done and they're going to write it down if only to do something else exciting.

Carl Franklin: You use the Reactive Extensions in building Simple.Data, did you not?

Mark Rendle: I did and then I got rid of them again.

Carl Franklin: So tell us that story.

Mark Rendle: So what I wanted to achieve was getting a connection to a database open and pulling everything else as quickly as possible and then closing the connections for database again because the database connections are expensive and they're not things that you want to look at for ages. So because I'm presenting the results of the query as an IEnumerable, people might sit there and go, okay, foreach, and now I'm going to do something that takes two seconds and then I'm to look around the foreach again. If you're doing that and there's a thousand rows out there waiting to be processed by that foreach, that's going to take a long time and the connections held open. So I want to kind of pull everything into a buffer and then present that as an Enumerable and Reactive Extensions has a fantastic way to do it and I guess you can basically say, okay, I'm going to turn it into an Observable and then I can just push the rows as I fire host from after the database into the Observable and then Reactive Extensions has two Enumerable methods which handle all this buffering internally with a dynamic queue and all this sort of thing and it works really nicely. The nice thing, well, as soon as the first row was returned, your foreach loops started enumerating so you weren't waiting on all that buffering to happen if you were selecting a lot of the rows back, but before you'd finished enumerating, the database connection is close and back in the pool and other people could use it or other threads could use it or whatever. So I was very happy, and then the Reactive guys has decided not to play nicely NuGet.

Richard Campbell: Oh.

Mark Rendle: And so they took off this strongly-named version assembly that Simple.Data was depending on and replaced it with another new one and everyone's code stopped working and I just thought, well, I can't have dependencies on projects that are going to do that kind of thing to me in my code and I actually release 1.0 today and they've kind of this 1.0, we're going to support this one, it's going to be around for a while, and so I could potentially look at taking that dependency back again, but at that time I threw my toys out of the paren and went "damn you."

Carl Franklin: Yeah.

Richard Campbell: So how do you go about doing testing against Simple.Data?

Mark Rendle: Mostly behavior test really. So I have an ADO adapter which works with IDb Connection and IDb command and that sort of thing, and I have a set of -- I'm still not sure whether they're Maxwell desktops, but they're basically a set of classes that will be an IDbConnection and a command and that sort of thing which I can initialize with in-memory schema. So, __________ 17:08 in the behavior test project and then I can just call my dynamic calls and so the database to open .products.finalbytype.orderbythis.skipthatmany.takethismany and then just verify that that created the right SQL and that proves most things and that's kind of quick. Every time I save the code, I have a mighty move setup and it just goes and it runs all those in-memory lightweight quick tests and then I also have specific tests for the SQL Server provider and the SQL Compact provider to make sure that things work when you run against natural database because obviously it's not much easier if they don't.

Richard Campbell: Right.

Mark Rendle: But yeah, they're off, they have 300 tests now I think and it's very hard to work out the coverage because the coverage tools tend to find it difficult to go from my dynamic syntax and work out where the code, part of the execution, is from that but I just kind of go. Everything I think should work seems to be working. So there maybe some slight issues in some places, but as long as I have no data at the front of it then I'm okay to just have hope for the best and fix things quickly if people do find problems.

Richard Campbell: The perpetual beta.

Mark Rendle: There are things that people turn up and I kind of go, oh, head slap moment.

Carl Franklin: That's a PWOP right there.

Mark Rendle: Yeah. Not the least of which was I spent some time really trying to optimize it a bit out of the Find by functionality because I was going up against Sam Saffron and Rob Conery and their Micro-ORM tools which are very, very quick doing that. So I spent a couple of evenings kind of going over the code and caching a lot of stuff and creating dynamic methods on the fly and that sort of thing. I got it really, really quick and I was like, yay, and it's within sort of 500 queries and it was within 400 of a second of those guys and then someone sent me a poll request a few days ago saying "You realize you're executing the query twice, I fixed that for you." And I went and to me it was right. So yeah, that was an interesting -- I sort of made all that effort to optimize this kind of micro-level and then just go. I just run that twice in for every single request.

Carl Franklin: It's a community at its best right there.

Mark Rendle: Yeah. I got a lot of -- I've been really impressed with the community on this. There are people who are just picking it up and using it and giving me lots of my speed by telling me "This is great. Could you just get it to do this? If you get to just do that, I'll go away." People have kind of pitched-in and written providers for the idea of adapter. So there's a guy, Bobby Johnson, who did a SQLite adapter, and another guy has done MySQL adapter, and yet another guy has done an Oracle adapter which was...

Richard Campbell: Nice.

Carl Franklin: Wow.

Mark Rendle: That was a big feature request. Lots of people are going, "If it supported Oracle, I'd use it," and Frank Quednau came along and he has written an Oracle provider for it and also submitted quite a lot of improvements to the code for the core project so I've given him commit writes on the central repository now so he just picks his things when he gets the time.

Carl Franklin: This portion of .NET Rocks! is brought to you by our good friends at Telerik. Hey, can you ever have too many free tools to compliment your development skills? I didn't think so. So our friends at Telerik are giving you now more than 30 free products for application development, automated testing, Agile project management and content management and we're talking free, free. Not a trial, not a demo, but free complete products supported by a community of over 440,000 developers at Telerik forums. From a free ASP.NET AJAX, ASP.NET MVC, and Silverlight controls to the free ORM solution and automated testing framework to free Agile management tools and content management systems, all these and more are available to you for immediate download at www.telerik.com/freestuff. Most of the free products can be use for commercial purposes and give you access to supplemental support resources such as documentation and forums. Go to www.telerik.com/freestuff now and take full advantage of the available free of charge products, and don't forget to thank them for supporting .NET Rocks!

Richard Campbell: Well, interesting. You're making an interesting statement about certain folks that started contributing to your project where they're apparently as committed as you are to it.

Mark Rendle: Yeah, which is with an important responsibility. Yeah, it's one of the things where I didn't start this to kind of go here's a huge thing that I want thousands of people to use. It kind of happened by mistake, but fortunately this kind of framework __________ 22:23 creating APIs. That's my favorite bit of programming on rubbish user interfaces. Fortunately, where I work we get people to kind of design the UX. We're very aware of what we can do and what we can't, and so if someone says they need a funky UI then we'll just get a part of it done and work with them on that, but I like kind of fiddling around with the nuts and bolts, and the data access, and streams and all that kind of good stuff.

Carl Franklin: I love that stuff too.

Mark Rendle: So Simple.Data to get kind of picked up just to the extent that it has so far and by now it's kind of huge or anything, but it's got a growing following. So these people are pushing it in directions that I'm finding really interesting to work on and follow through on which is really neat.

Richard Campbell: You've talked through various SQL adapters. What about the NoSQL type data sources?

Mark Rendle: I mean so this came from the Ruby Data Library which Ruby has too, ORMs, Active Record, and Data Mapper. An Active Record is very much a Relational Database ORM. That's what I do. Whereas Data Mapper has basically said what I'm going to do is I'm going to take all this stuff over here and I'm going to turn it into hashes in Ruby and I'm going to pass them through to your data and you give me hashes back and I'll turn those back into objects. So I just got this doorway, or bridge, or whatever over which only hashes pass and Simple.Data takes the same approach with its adapters except everything is a dictionary of string and object and that means that it will map very nicely to all sorts of things as well as relational databases. The only one that's kind of helped in the wild and getting supported is MongoDB so far but someone was talking about doing one for counts, someone was talking about doing one for retests, and I'm going to be doing one for Azure Table Storage.

Richard Campbell: Nice.

Carl Franklin: Wow.

Mark Rendle: This is fundamentally if you get into joins and grouping and all this sort of stuff, there are things that relational databases are extremely good at and it's probably going to be quite hard to make Simple.Data do those things against every possible data store.

Carl Franklin: Right.

Mark Rendle: I've written that for registry which was evil and I have burned the code.

Carl Franklin: Yeah.

Mark Rendle: Sorry.

Carl Franklin: I was going to say rest in peace.

Mark Rendle: Yeah, yeah. Rest in a lead-lined coffin that the customer will see for that one. We do not want to be running select statements against XP local machine.

Carl Franklin: Oh.

Richard Campbell: What could go wrong?

Carl Franklin: Yeah. Please wait. Do you think that when you're implementing the Azure provider that you will be able to maintain absolute backward compatibility with the interface in terms of what the user uses?

Mark Rendle: Mostly. There are things that you can do in SQL, like you can say it grew from this and ordered by that the Azure Table Storage doesn't support in the same way and so the choice basically is I go, okay, I'll do what I can with filters and that sort of thing and then I'll just stop pulling data back and I'll do the rest of it in memory, but I think what I would like to end up with is a kind of a fairly high/low and lowest common denominator that can say, okay, find by will work, find all why will work. These couple of bits of query will work, the delete will work but then there's kind of a red line and everything the wrong side of that red line is provide a row over that to specific and so you'll mileage may vary.

Carl Franklin: Yeah. Do you think your users will come to expect that? I mean for example, do you expect that somebody moving from SQL Server to say Mango will expect all those features of the relational database in Simple.Data?

Mark Rendle: I think we're on the risk of leading them to expect that, yeah.

Carl Franklin: Yeah. I would say that you would have a separate interface for those just because they're two different animals and they're used quite differently.

Mark Rendle: Yeah, and I see your point. One of the nice things with keeping it common is that as your table storage has excellent things on, actually it can't do everything what the SQL database can do. A SQL database can do everything that Azure Table Storage can do. One of the issues that you have when you're developing for an Azure project is the table storage isn't there when you're in dead mode and you can use the local development fabric which has an emulator that runs on top of SQL Express, but if you run your website outside of the Azure compute emulator and storage emulator, then that's not there either and you can't access it and Simple.Data would give you the ability to say, okay, I'm going to create a SQL Compact database or even an in-memory SQL-like database that matches what I'm doing in table storage and just by changing the connection string in the config file I can run that on IIS Express or in Cassini or in my local IIS and it will work in the database and then I can deploy without making changes to the actual application code or the Data Access code and it will run against as your table storage. That's one of the things that I think is quite attractive to me because I do a lot of Azure development. So being able to achieve that parody between two different data storage systems is something that I would find quite useful.

Carl Franklin: I see that you fancy Nancy.

Mark Rendle: I do fancy Nancy, yes.

Carl Franklin: I've always wanted to say that. What is Nancy?

Mark Rendle: Nancy is a Sinatra-inspired low ceremony web framework and the two guys who do most of the work on them, you see I like to refer to it as the super duper happy path to web development which I can totally see.

Carl Franklin: Yeah.

Mark Rendle: So in the same way that I've kind of written Simple.Data because I think there's an awful lot of craft associated with things like Entity Framework and NHibernate, Nancy strips away all the craft associated with things like MVC frameworks. So in Ruby, well, you've got Rails and Rails is huge and is opinionated and it says you should use this and that and the other and I'm just going to assume that what you're creating here is a REST API over a relational database, I'll generate all that for you and God help you if you try to change anything. What Sinatra does, and therefore what Nancy does, is it says I'm going to assume that you're writing a Web application and therefore you're going to need to respond to URLs. So I'll help you do that and then the rest of it, Data Access forms layout, you do that however you want to do that and we'll just take care of parsing those URLs and form posts or date query post or whatever it might be and we'll turn those into dynamic variables for you. So it sits really, really nicely with Simple.Data and quite a lot of people seem to use them together and they do work quite nicely together.

Carl Franklin: Wow.

Richard Campbell: It does seem like there's lightweight-focused sort of minimal movements going on that you seem to be at the center of it in some ways.

Carl Franklin: Definitely.

Mark Rendle: Yeah. I think that's good. I think in a lot of projects that I've seen, and particularly I do a lot of work with companies who have got sometimes immature web software that they're looking to migrate to Azure and there's so much code in there that do basically just boilerplate and it's repeated over and over and over and over again.

Carl Franklin: Right.

Mark Rendle: I mean I see people who have tied themselves up in knots trying to implement Dependency Injection which is the exact opposite of appointed Dependency Injection. I've worked on projects where they had basically taken LINQ to SQL back in Visual Studio 2005 and they needed about three tables after the 300-table database but they just generated the model to the entire database store procedures to be used a whole lot. So you have this enormous ADMX file over DBML file and every single thing in the database was modeled as a class and this was just so they could do authentication and display all their information. So I think a lot of people are reacting against that and going how can I, in a small amount of code as possible, get across the essence of what I actually want my application to do?

Richard Campbell: Right.

Mark Rendle: I think in C#, LINQ introduced this idea of declarative programming particularly with the LINQ syntax itself rather than having to say this is how I want to loop over this, and this is how I want to aggregate that, and this is how I want to do my ordering. You just went "Can you do this for me, please?" And the compiler came along and it's not necessarily smarter than you but it won't take shortcuts, and that got people into this state of mind where they're going, well, why should I have to write all this code to tell my application how to do everything? Why can't I just tell it what to do? That's something that the C# team is kind of progressing towards. I don't know if you have heard Jon Skeet's talk on how to do Async Await Keywords.

Carl Franklin: As a matter of fact, I did a dnrTV with him in Norway where he showed essentially what you're talking about. He didn't use it. He showed how to do it in C#.

Mark Rendle: Yeah, yeah.

Carl Franklin: Like how he thinks it could be done just as an exercise in how to provide an async keyword. That's brilliant.

Mark Rendle: Including a bout with the compiler syntax.

Carl Franklin: Right.

Mark Rendle: You kind of end up about 800 levels of indentation where they sort of wait and to do and we've got the Task Parallel Framework and so we can say whatever this task can do they sort of continue with that and they continue with something else, and they continue with something else. But that very quickly gets all kind of nested and Node.js, which is enormously popular at the moment, that has the same thing. It's all very well kind of going. It doesn't block, it never blocks anything handled through in a callback, but by the time you got the request in from the web browser and then you pass the arguments into strings and then you've authenticated the user and then you've got the user's shopping cart, the tiles, and then you've added something to the shopping cart and then you've updated the stack control that is not in stack anymore and then you finally return back, you've got 16 levels callback and a nice async/await syntax just does all that for you at one level of indentation.

Carl Franklin: At Franklins.Net right now, you can get a DVD with over 11 hours of Billy Hollis on Silverlight 4.0 or 14 hours of Sahil Malik on SharePoint 2010 each for only 695. Order online at www.franklins.net. Are you looking to change jobs? Infusion Development has offices in New York City, Toronto, London, Dubai, and Poland. Infusion has hired a whole handful of happy .NET Rocks listeners. Contact me for an introduction at carl@franklins.net.

Speaking of simple standards, you're supporting the OWIN movement? What is that, the OWIN?

Mark Rendle: Open Web Interface for .NET and yes, I haven't been involved as much recently because I've been doing all sorts of things and so that kind of took a back seat but I'm the one who's responsible for the hideous delegate signature.

Carl Franklin: What is it exactly?

Mark Rendle: OWIN, it's a reaction to the fact that Microsoft's HTTP Request and HTTP Response objects failed and have internal protective constructors.

Carl Franklin: Yeah. I don't know.

Mark Rendle: You can't make one.

Carl Franklin: It always seemed like a silly thing to see all that class to me.

Mark Rendle: Yeah, it is and it's a silly thing for something like an HTTP Request where it's a classic example of something where the implementation should be entirely separate from the contract. Why is there no HTTP Request? And this is something that Microsoft I think they are getting better at it, but it's talking a long time in going back and reapplying these sorts of things, things that are something that they're finding it hard to find the time to do. So writing a testable application, even an ASP.NET MVC which is designed to be more testable, it generates test projects for you using either MSTest or I think it's called MbUnit now because Phil Haack liked that, but if you have to go in and hand crank all that code that it generates like Request tabs and Response tabs and everything else. It's back to that thing of writing loads of boilerplate code to do something that should essentially be very, very straightforward. So OWIN is a project that kind of came about with some guys who are all writing their own web servers and web frameworks and just went, "If we had a common interface between the web server and the frameworks, then we could choose your framework on my server, or you could choose my server in his framework, or my framework in his server and that would be really neat." So they've kind of formed a working group and are attempting to define this as a standard and my contribution was to pitch off and say "But you're using interfaces, and if you use interfaces you have to have assembly interfaces defined in; whereas if you use Action and Funk delegates, then you wouldn't need those things." So the whole thing is being kind of built around Actions and Funks to get passed backwards and forwards and the result being that there is a kind of reference implementation of the middleware that attaches the web server to the framework or to the multiple frameworks and can inject the middleware and so on, but that doesn't have to be the only implementation. If someone thought they can do a better job, then they're perfectly at a liberty to do that. It's very cool.

Richard Campbell: It is very cool and it's an interesting way of thinking about the problem as well. It's just that I feel like we're also trying to get close to the metal all the time. It's so hard to actually alter the stream of data coming and going from a web server because you've got all this abstraction and pipelining.

Mark Rendle: Yeah, yeah, absolutely and you just sort the thing. By the time the request is coming through IIS, ASP.NET has managed to get a look at it and turn it into an HTTP Request object and serialize the first part of it into a dictionary and read all the cookies and all this sort of stuff and then it passes it off to ASP.NET MVC and MVC looks at it and it tries to work out which controller should be handling it, which action should be handling it, and have more values off the request and turn them into parameters, and then finally if you're lucky and they're all sending HTTP modules waiting there to have their turn, you can start doing something. If on the web servers they are trying to server 2-1/2 thousand requests per second per call, that makes it hard.

Richard Campbell: Yeah. That's unlikely, it's what that is, but do you think this is primarily a performance issue or a code maintainability issue?

Mark Rendle: You mean the movement towards the metal?

Richard Campbell: This whole idea around OWIN as well. You know, I have these debates when we're talking about doing stuff like inserting into HTTP headers and so forth about where do we do this? Do we build a module? Do we subclass it? I mean there are so many different ways to go about things when you're living in the IIS, ASP.NET MVC space. When you strip a lot of that out it gets a heck of a lot simpler. So for the call minutes, it makes a lot of sense, but from a performance point of view, I just don't know if that's the motivator because there are other ways to solve that too.

Mark Rendle: Oh, yeah. I don't think performance -- I mean in these days of Cloud service and stripped disk and memory and processors, scaling out and scaling up are not particularly prohibitive anymore, but I do think people like the idea of having lower level control over what's going on.

Richard Campbell: Sure.

Mark Rendle: I do think -- I mean the code's maintainability is that less code between the TCP socket and your framework. It feels naturally like that's a good thing and it's just less code in your application, that's less code to go wrong. If everything is handled through interfaces or delegates, then at any point in that pipeline you can inject something in a test environment; and as far as everything beyond that point is concern, that might as well have come from a TCP socket so you can create a dictionary of headers and you don't have to patch in everything else. If you're just testing something in your header handling, then you can just bring in a dictionary of strings string and pass that through another handler. Whereas, if you want to just do that with an HTTP Request, you can get funky and go into Reflection and call constructors that you're not supposed to, but then again, then you've got to maintain all that Reflection code and worry that Microsoft is going to change the way that works internally on the next race and then suddenly your test break.

Richard Campbell: Yeah. I guess this is the whole dance here, it's wrestling on what dependencies you're willing to take and how much you trust the underlying code and we're all at the place now where it seems like where when you dig into the stuff, when you bump into a sealed class it makes you run the other way.

Mark Rendle: Yeah.

Richard Campbell: You seal that, I can't trust it.

Mark Rendle: Why did you seal it? What are you hiding?

Richard Campbell: What are you hiding?

Carl Franklin: Right.

Mark Rendle: Or alternatively, I mean the standard case in sealing that class is so that someone else can't come along and create a derived copy of that class and then post it back up into the framework and calls mischief or just break it accidentally through ineptitude, but I think there's now a generation of programmers on .NET who are going, "Please don't protect me like that. I don't want to be protected like that. I'm a big boy."

Carl Franklin: Yeah.

Mark Rendle: And he is now, but I can choose whether I'm going to run with the scissors and you sealing that class just makes me want to go and rewrite the whole stack from the ground up with an unsealed version of that class. So that's what I'm going to do, bye.

Richard Campbell: Is that the real description of OWIN right there?

Mark Rendle: I think certainly some of the people were involved, that's their attitude.

Richard Campbell: Sure.

Mark Rendle: I think there is a growing movement of people who, they're not anti-Microsoft.

Richard Campbell: No.

Mark Rendle: And they're not anti-.NET, and I include myself in this number. I would like to see .NET, as a cool framework shrink back down.

Carl Franklin: To say Silverlight's ass?

Mark Rendle: So WPF and ASP.NET and all these sorts of things, those aren't part of what you call a cool framework. You could take just the system and System.Network and System.I/O and those few dozen or so cool namespaces and build anything you wanted if you sort of plugin an OWIN web server, and I'd say if you want to keep ADO.NET because no one wants to rewrite that and the Simple.Data needs it so I don't want to be rewriting that, but Mono doesn't come with WPF; it comes with the GDK shop so even something as kind of low level as a client development framework, it's possible to do that from the primitives. So I think a lot of people would like it if .NET become a bit more like Java and JVM or Ruby or Python where they just have the primitives and then all these other things are available to install if they want to, but if they don't, then they can just build things from those primitives.

Carl Franklin: So Mark, what's next for you in Simple.Data? What's on your horizon?

Mark Rendle: I'm very, very close to 1.0 and I've got to do __________ 48:24 for the queries, I think that's the last feature on the queries, do a bit of work to support the guys who are writing that against non-SQL databases and some code cleanup and some optimization work and then I'm calling it 1.0 and then I'm going to walk away from it for a while. So I'll support it. So if you're out there listening and you're going, "Oh, no, he's going to dump it," I'll kind of make sure it carries on working and so forth. I'll probably write a couple more adapters for it like your adapter, but my next plan is to port Node.js to F#.

Carl Franklin: Nice.

Richard Campbell: There seems to be a ton of energy around Node.js. There are all kinds of talk about porting it to .NET.

Mark Rendle: Yes. I mean Microsoft is working with Joint now getting a native build of mode to run on Windows which is great and fantastic. I think having Node on the same server is kind of SQL Server's 2008 R2. It has a lot going for it, but at the same time, like I said earlier, JavaScript, it invented callback type model. It's very good for doing that kind of non-blocking, for expressing that non-blocking code but it doesn't necessarily do it in a particularly readable maintainable way. Whereas F# already the async keyword and so if someone, and I can't remember the name of the blog but I will try and find a link to it and send it on to you, guys, showed an example of doing something and Node.js and then how that would look in F# using async he would...

Richard Campbell: Oh, yeah.

Mark Rendle: He would basically async this, async that, async the other, async right back to the string.

Carl Franklin: Nice.

Mark Rendle: Whereas Node.js is getting itself all that wrapped-up braces and parenthesis and the weird thing that it does where you have to create a function and then put .call on the end of it and whatever else even if you use CoffeeScript just to lending out to these multiple levels of white space indentation, whereas, F# can do all of that in a much more complete and tidy way and it also have access to all the .NET framework awesomeness and you can sort of hook into a Task Parallel Library on ADO.NET and all those sort of the lower level bits and pieces. So I think node.fs is probably not that difficult to at least to a proof of concept

Carl Franklin: Cool. Well, thanks very much for hanging out with us this hour and it sounds like great stuff, and like I say, get some sleep.

Mark Rendle: Yes. I think I need some sleep. I think I've done kind of a 99% of the talking there so I apologize.

Carl Franklin: Oh, no. That's good. That's why you're here.

Mark Rendle: Okay, good, but not a ton.

Carl Franklin: Thank you, Mark, 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! episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.