



[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 #461*  
(Transcription services provided by [PWOP Productions](#))



**Stephen Forte on Data Access Options**  
**July 7, 2009**  
*Our Sponsors*





**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! Take those bottle rockets off the cat 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 #461, with guest Stephen Forte, recorded live, Tuesday, June 30, 2009. .NET Rocks! is brought to you by Franklins.NET - Training Developers to Work Smarter and now offering DotNetNuke video training with Chris Hammond from Engage Software on DVD, dnrTV style, order your copy now at [www.franklins.net](http://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](http://www.telerik.com), and by CoDe Magazine, the leading independent magazine for .NET developers, online at [www.code-magazine.com](http://www.code-magazine.com). And now, the man who truly loves animals, especially with a good wine, Carl Franklin.

**Carl Franklin:** Thank you very much. Welcome back to .NET Rocks, Carl and Richard here for your listening pleasure. Hey Richard, what's up?

**Richard Campbell:** Ah, you know, nothing but good stuff.

**Carl Franklin:** No rest for the wicked.

**Richard Campbell:** That's what they tell me.

**Carl Franklin:** Hey, let's just include Forte in on the intro. Hey man, what's happening?

**Stephen Forte:** Not much guys.

**Richard Campbell:** You might as well be a co-host. He's done, what, a dozen of these things.

**Carl Franklin:** Yeah, right, you're honorary. If Richard ever keels over I'm calling you, man. You're the next in line.

**Stephen Forte:** As long as I don't have to move to Canada.

**Richard Campbell:** Nice.

**Carl Franklin:** Yeah, and you'd miss some great pizza at Pepe's but we won't go into that now. Hey, it's time for Better Know a Framework.

[Music]

**Richard Campbell:** All right.

**Carl Franklin:** All right. So, Richard, do you know the difference between a `system.timers.timer` and a `system.windows.forms.timer`?

**Richard Campbell:** I do not know the difference. Please enlighten me.

**Carl Franklin:** Well, I'll tell you.

**Richard Campbell:** All right.

**Carl Franklin:** There are two timers in the .NET framework. You know what a timer is from the old VB days, you drag a timer...

**Richard Campbell:** Yes, I think it raises an event after a certain amount of time has passed.

**Carl Franklin:** Exactly. So the Windows Forms timer is obviously for use in Windows Forms only.

**Richard Campbell:** Right.

**Carl Franklin:** And its accuracy only goes to 55 milliseconds.

**Richard Campbell:** Only.

**Carl Franklin:** And it's single threaded, yeah, single threaded.

**Richard Campbell:** Yeah.

**Carl Franklin:** So, you actually set the interval in milliseconds, but 55 is as low as it goes, but if you set it for 1 millisecond, it doesn't matter, 55 milliseconds.

**Richard Campbell:** Right.

**Carl Franklin:** If you use a system timer's timer, you have much greater accuracy, but it's multi-threaded so that timer event is going to happen on its own thread. So, if you expect to change any Windows UI, you can't do that without first using the `ISynchronize` interface to do an `invoke` of a delegate blah bitty blah de blah...

**Richard Campbell:** This is an area especially for you that whole asynchronous behavior thing.

**Carl Franklin:** Yeah. So basically, when it comes down to is if you want a timer on a form and you're going to update some UI, use the timer, Windows Forms timer but if you're going to do something that doesn't access the UI, you can use a system timer's timer which is in your components on the Windows Form, it's just that you will not be able to like access your controls and do any UI stuff.

**Richard Campbell:** Right and how accurate is this system's timer?

**Carl Franklin:** It's to a millisecond.

**Richard Campbell:** To the millisecond.

**Stephen Forte:** And does it do military time.

**Carl Franklin:** Does it do military? I mean can you specify? What do you mean military time?

**Stephen Forte:** Twenty-four hour clock.

**Carl Franklin:** It's a timer, you're telling them milliseconds. What do you mean does it do military? That's the dumbest question I ever heard Mr. Forte.

**Stephen Forte:** I have my computer set to military time, will this clock work?

**Carl Franklin:** You set it as an interval of milliseconds. You don't tell it at 4:00 p.m., at 0400 hours I want you to fire.

**Richard Campbell:** How many milliseconds is that?

**Carl Franklin:** Yeah, exactly. So if you're in the military this is not going to work for you, I think that's what you're saying.

**Stephen Forte:** I heard there's an open source implementation called iTimer...

**Carl Franklin:** Yeah.

**Richard Campbell:** Oh, no.

**Stephen Forte:** And it works to 42 milliseconds.

**Carl Franklin:** Who invited him?

**Stephen Forte:** I don't know.

**Carl Franklin:** So Richard, this is the time for you to read an email.

**Richard Campbell:** I have an email and probably one that Steve will appreciate. "Hi, Carl and Richard, just wanted to say a quick hello, well overdue since I've been listening to your show from episode one."

**Carl Franklin:** Oh.

**Richard Campbell:** "Not only as .NET Rocks! both informative and entertaining but it also shows me that there are others out there who like to talk about .NET and I'm not just weird. One of my rules is two times is a coincidence but three times is a pattern and I practice that faithfully with your show. If a certain technology gets mentioned three or more times and I don't already know about it then it's time for me to get reading." Wow, so I should pick really random technologies and put them on the show three times just to mess with him. "However, I'd like to bring up a recurring topic that you probably wish you could put to bed by now, specifically the question of about identity versus GUID as the primary key in a table. There are three key points that I'd like to raise, one, the reason I would use GUID, a big random number, as the primary key is because I can give my objects a GUID identifier in the domain layer and link them by GUIDs without requiring a trip to the database and that once I have this unique identifier, why should I add yet another unique identifier? Two, the point that an identity column primary key will improve your right times since the primary key is clustered, needs reconsidering. A clustered index is intended for a right time penalty with an expectation of a read time optimization. It's only benefits where the table will be read in the order of the key, ascending or descending, or when restricting by the range of the key. Since as we both agree that the identity value should not be exposed to the business user, then it is unlikely that the table would ever need to be read in order of its identity but rather by the keys such as name, date, location and other external identifiers. Actually, if you turn the clustering of the table off and your table has only occasional leads, then your write performance will be comparable to that if you have a clustered identity primary key because in both scenarios, the rows will tend to be added to the end of the data."

**Carl Franklin:** Aha!

**Richard Campbell:** "And three, when your value of inserts is such that you need to worry about whether your primary key is random rather than sequential value, then you probably better restructuring your data for write optimizations such as splitting the writes between two or more tables or even between two or more databases and the amalgamating the data later. However that said, when it comes to large reporting databases especially when you're setting up facts and dimensions, nothing links data together faster than an integer and so while I love my GUIDs and I did say I was weird, identity certainly has its place and I better stop now because I can go on about this sort of thing for hours, thanks again, keep it coming and how about penciling in an Ozzy trip someday, for real."

**Carl Franklin:** Ooh...

**Richard Campbell:** That's from David Swan, Down Under. David, thanks for your email. We'll send you out a mug. You're totally wrong but I'm okay with that.

**Carl Franklin:** Okay.

**Richard Campbell:** You know...

**Stephen Forte:** You're arguing that you should be using GUIDs, Richard?

**Richard Campbell:** He's pro GUID, there's no two ways. He's pro-GUID and I mean his first reason, the idea that, "Hey I can generate GUIDs against my objects and I don't have to go to the database for the identifier is a fairly reasonable one in the sense that the probability of a GUID being already used in a database after generated is almost zero. So you do have that ability to create unique identifications in an application independent of the database whereas if you use a traditional identity, I have to go to the database to get it.

**Carl Franklin:** Still doesn't make it good idea!

**Richard Campbell:** Well...

**Carl Franklin:** I'm just kidding, I don't know.

**Richard Campbell:** The second point was the one that I really struggle with which is this whole thing about clustered versus non-clustered. Look every table is written in order no matter what. All that happens when you cluster it is you tell it what the order is.

**Carl Franklin:** Yeah.

**Richard Campbell:** And the real issue here is and this is what Paul Randall was talking about is that if you write sensitive, if the rate of the writing matters then the non-sequentialness of GUIDs causes harm to your database, to your table. It actually really makes it hard to write this out, it slows down the rate of write but if the write rate isn't important, it's not a big deal.

**Carl Franklin:** The write rate?

**Richard Campbell:** The write rate.

**Carl Franklin:** Is the write rate right?

**Richard Campbell:** It doesn't matter.

**Stephen Forte:** Unless you're clustering something other than your primary key which I have argued for decades, ever since I was in grammar

school I would always say argue over some, cluster your database index to that table over something you can do between clause and your where clause or an order by and he brings up a good point, is that when you are going to search a query you usually you usually are going to say, "Show me customers between customer ID x and customer ID y because in theory, that's unique and no one knows those numbers because they're irrelevant to the application but you might search between order date and I think that would make a better candidate for a clustered index. So you're right Richard when you're clustering on the primary key but I think that he has a point when you're clustering not by the primary key.

**Richard Campbell:** And the number of scenarios where you're actually going to cluster by something of, the big thing about clustering on the primary key is if you're doing unique row pulls, so if you're going after a customer, you're going to pull the individual customer and the clustered index is great for that other than pairing the write, the reads are awesome.

**Stephen Forte:** I actually got it in reverse, if you're doing unique pulls, you don't want a clustered index as much, you want a clustered index if you're doing a between clause. You get a far better response time when you're doing a between and you're between is on a clustered index. You're looking for one thing in an index? Having a clustered index is not going to really be all that much advantageous for you.

**Richard Campbell:** It doesn't make much difference compared to a non-clustered index but you know that it's famous thing either way. Clustered versus non-clustered just where it's going to be stored? So the C time is essentially the same it's a question of when it actually fetches the data what does it need to fetch where I see the clustered index having an advantage and then a single row pull makes sense because if I'm pulling the whole customer then the fact that I don't have to go read into an index, get my row identifier and then go to table to get it, to get the actual data and if that's slower than if I use the clustered index as I'm pulling the whole row.

**Carl Franklin:** Welcome to another stellar episode of SQL Rocks!

**Stephen Forte:** Well, I think they should have actually a way to end this debate once and for all and the way you end this debate once and for all is rename a clustered index and remove the word "index" from the title because in reality it's not an index, it's just how it's physically sorted on the database. All right, so a clustered index, technically speaking, is not even a real index, according to SQL Server. So if they change the nomenclature, it might change the debate.

**Richard Campbell:** Yeah, it's still treating an index as part of a query plan so it is an index but I admit it's a different thing and it can't be compared easily. If you're going to end this debate I tend to agree with Paul Randal and he says GUIDs in as clustered index, bad.

**Carl Franklin:** Okay and with that, that ends the mail reading.

**Richard Campbell:** Nice, yeah.

**Carl Franklin:** Well...

**Richard Campbell:** First 10 minutes of the show, we're already into a database debate and it, well, we were going to talk about data access anyway.

**Carl Franklin:** Sure. We just, I guess we don't really need to introduce Stephen, I mean he's been on the show a million times before, just blog Stephen Forte and you'll find out all about him but he does have a bio. What are you currently doing Stephen?

**Stephen Forte:** Well, I'm currently doing a lot of things, I'm standing in my living room talking to you guys on the phone. What I'm doing full time like as in employment speaking, between climbing mountains, I'm the Chief Strategy Officer of Telerik, the ASP.NET and Windows Forms and Silverlight and WPF tools vendors amongst other thing, .NET tool vendors.

**Carl Franklin:** That's great, of course they're huge sponsors of the show and how are they doing? We don't talk to them much, they're out there over on the other side of the planet.

**Stephen Forte:** They're doing well, just geared up their mid-year release has been out and successful and every one seems to be happy so they're doing pretty well.

**Carl Franklin:** That's great. That's great to hear. So we're talking about, what are we talking about Richard?

**Stephen Forte:** Well Steve put together this great demo a while back where he showed all the different new technologies that Microsoft's using for data access, Entity Framework and Astoria, and so forth...

**Carl Franklin:** Oh, all right.

**Richard Campbell:** You know this is an interesting conversation because I feel like the way we get it data from Microsoft, this conversation is very confused

right now. So I wanted to sort of debate through all these different methods based on Steve's research.

**Carl Franklin:** Okay. So you're basically trying to bring a little common sense to people who argue, you should use this, when you should use that, when this one sucks, stay away from that one, don't use this one. That's what you're really trying to get some clarity around, right?

**Stephen Forte:** Well, one of the things that I'm looking for is not necessarily saying, I'm trying get clarity more around the ones that say blanket statements like, this sucks, never use this or always use that and I feel that the situation warrants, many different situations, more different technologies. There are so many different things that we can choose from out there that I think sometimes, I think, quite frankly the market place is confused.

**Carl Franklin:** Yeah. I never pay attention to people who say never use this or never use that.

**Stephen Forte:** Or always use this or always use that...

**Carl Franklin:** Yeah.

**Stephen Forte:** ...as well, I think, by converse.

**Richard Campbell:** But by same token, I don't get the sense that Microsoft has a grand data access strategy that they have, they've said, "That we're going to need four ways to get into data and you're going to build this one and you're going to build that one and you're going to build that one." I feel it's more like these guys have ideas and they try them and it's up to us if to figure out how to use them; or if we even should.

**Stephen Forte:** Yeah and I think this is part of Microsoft greatest strength and it's also, conversely one of their greatest weaknesses and it goes back to the earliest days of how Microsoft was run where you would be almost an entrepreneur inside of Microsoft. You'd work on something kind of nights and weekends, you'd prototype it; it would go to Bill Gates or somebody else usually Bill Gates and they would grant you a team and a budget and say, "This is great." And then the converse will happen and it wouldn't work so the problem is we would get many different ways to do the same thing. Think back 10 years ago, we had Form packages for every day of the week, right? We had Visual Basic forms, J++ forms, I don't want to even list them all. There were Outlook forms, there were Office forms...

**Carl Franklin:** Yeah.

**Stephen Forte:** There were C++ forms, J++ forms, you name it. So that was a great thing because there was a lot of innovation but it was also a bad thing because it confused the market place and then of course what happens is a great kind of consolidation happens and the best from each package kind of gets consolidated and that's what happened about eight, seven or eight years ago when the .NET framework really kind of took hold and took the best from all of those packages and that's kind of happening in data right now where it's a great strength because there're lots of data access methodologies being produced by Microsoft, right but of course at the same time that's a weakness because for the guy that just wants to get his application out the door, spending a lot of time just figuring out which one do I use.

**Richard Campbell:** So Steve, what's wrong with ADO.NET? Why wouldn't we just use that?

**Stephen Forte:** Well I think that nothing particular stands out as absolutely wrong with ADO.NET but what happens is applications or maybe not applications but the way we've developed applications has started to evolve...

**Carl Franklin:** Well and they're getting big.

**Stephen Forte:** We move into a, yeah?

**Carl Franklin:** I was going to say they're getting big, they're getting big and writing all that same ADO.NET code over and over and over again for, to do all the same cruddy stuff is just not something that people want to do. You spend a lot of, if you have an application with lots of tables and lot of data and stored procedures and things that you've got to write wrappers around, it just screams for some sort of higher level tool and I think that's what you're talking about here, these tools and technologies that we're using are tools that are higher level to abstract away some of the menial work, of accessing that stuff.

**Stephen Forte:** I tend to agree and disagree with you. I agree with you that we're doing a lot of repetitive things and I'll come back to that in a second, where I disagree with you slightly is that applications are getting big, applications are no bigger than they were 10 years ago.

**Carl Franklin:** I suppose you're right.

**Stephen Forte:** We are just adding different services, they're getting maybe a little more complex because we have to talk asynchronously to things and there's cloud and there's web services and all that kind of stuff.

**Carl Franklin:** Yeah. That's really what I meant, it's complex.

**Stephen Forte:** What's interesting is that I think that the way we've developed these applications that are maybe a little more complex is they definitely evolved. We're looking at things more having a domain layer, where that domain layer speaks about the application's business logic and core domain logic and now what happens is that developers are kind of looking at having a domain layer then that domain layer has to talk to this layer of stored procedure calls or TSQL calls and I think quite frankly, there's been the movement to kind of, as you just said abstract that away.

**Carl Franklin:** Yeah.

**Stephen Forte:** And move one layer of abstraction higher than the core ADO.NET stacks. So, I don't think there's anything wrong, necessarily, with ADO.NET but I do think that applications are evolving kind of faster than ADO.NET has evolved and that's why you see a newer generation of data access kind of coming out.

**Richard Campbell:** Is there also this whole angle on feeding data to Silverlight with a driver to the new data access methods?

**Stephen Forte:** Well definitely, because Silverlight does not allow you to talk to, does not allow you to have ADO.NET in it's applications.

**Richard Campbell:** Right.

**Stephen Forte:** So Silverlight and in Azure you can't just directly communicate with the back end database. It has to be done asynchronously through a service which is the best practice but also something that developers are trying to get their hands around. So if you were to do that with ADO.NET, you'd have to do something ugly like have an ADO.NET service, so to speak on a server. Use WCF to wrap up the result with a data set and developers seem to have a love-hate relationship with data sets and then kind of send that down to the WCF service to your Silverlight client and then work on it there.

**Richard Campbell:** Yeah.

**Carl Franklin:** So then the question becomes when all of these technologies, first of all, what's the list? You obviously have a list of these technologies, let's just list them off.

**Stephen Forte:** Oh, the list is pretty numerous, I mean first there's this is plain old ADO.NET, okay,

right that, that's the first on the list and then there are the Microsoft offerings which would be the LINQ to SQL Project and also the Entity Framework Project and then there is also the whole kind of third party ORM technology out there which the entity framework more or less would probably be considered part of. So the whole ORM category, you have open source versions, you have commercial versions and all the like.

**Richard Campbell:** Right. What about Astoria?

**Stephen Forte:** Well, Astoria is an interesting case because Astoria is not necessarily a data access technology in so far as it's RESTful service built on top of your particular applications. So you could go in and create a bunch of just plain old, normal CLR objects and just have regular collections. You can just code it by hand, you get a person and then a people collection and you could expose that to Astoria as a restful service and those objects only live in memory on your application, don't interact with any kind of database. However, what makes Astoria pretty interesting is that it can fit of top of the Entity Framework or third party ORM's or even LINQ to SQL. So Astoria is even one layer higher of abstraction than what we were just talking about and what's nice about Astoria is it really handles the whole Silverlight, Azure question for you pretty nicely because you can wrap up data in this Astoria service and using the RESTful protocol you can go in and grab that data then work with it on the client.

**Richard Campbell:** I thought that was a very interesting angle on that. I guess with the official name Microsoft Data Services like you know you've got a bad name when people use the code name to remember what the product is...

**Stephen Forte:** Well I think Microsoft's proposing that we go back to calling Windows Vista, Longhorn for the opposite reason but...

**Carl Franklin:** Here's an obvious one, LINQ to SQL versus Entity Framework?

**Stephen Forte:** An obvious one in so far as what, I mean there's, what should developers choose?

**Carl Franklin:** Yeah.

**Stephen Forte:** Or what...

**Carl Franklin:** When would you choose one over the other? I mean you've heard the LINQ to SQL sort of dying because the Entity Framework sort of took over that space. What, I mean is there, what are the differences and when we'll we use over the other?

**Stephen Forte:** Oh yeah, here are the key differences right? So LINQ to SQL connects only to SQL servers and this was probably what Richard was referring to at my TechEd demo when I showed the SQL produced by NHibernate, Entity Framework, LINQ to SQL and other third party ORM's and things like that and LINQ to SQL does the best job because it's hardwire to SQL server.

**Carl Franklin:** Best job meaning fastest most performant?

**Stephen Forte:** Not necessarily fastest, most performant unless you're dealing with really complex TSQL but the cleanest most efficient piece of TSQL.

**Carl Franklin:** Okay.

**Stephen Forte:** It's definitely produced by LINQ to SQL. LINQ to SQL does not support Oracle and other databases and LINQ to SQL often will support one to one table mapping. So it's a pretty limited feature set. However, if you are working on a small application or forget small application, you're working at an application whose requirements only require you to deal with the constraints I just gave you, one to one table mapping and SQL server, LINQ to SQL is a fine alternative, it's a fine thing to use. Alternatively, the Entity framework and just about every other third party ORM out there will communicate to multiple databases and will also give you the opportunity to not just do one to one mapping and a lot of ORM's and now the Entity Framework 4.0 which I find funny, I think they're counting, yeah they went from one to four because I think they were afraid name it three because three has always been the magical number at Microsoft so I'm not sure why they named it four.

**Richard Campbell:** Four has got to be better than three right?

**Stephen Forte:** It's got to be three plus one. In reality they named it that way of course because the Entity framework is part of a .NET framework 4.0 but it's good to get a good shot going on that one but the Entity Framework and most ORM's will allow you to actually start with a model and reverse engineer that to a database, something LINQ to SQL cannot do and I actually have a problem with ORM's and the Entity Framework in that respect because they force you to think about the debate between, should we do model first, meaning should we just have built all the domain objects and build our model click a button and then that would generate the database for us automatically or in the other side of that debate, you have the database people that say, "Ooh, we should definitely do the database first click a button and it forms all your class objects and I kind of say both are wrong, you need to do that some kind of collaboration. You need to build the database, you need to build the

object layer and they're going to need to talk to each other and that's kind of where those, ORM's are all evolving into that particular direction.

**Carl Franklin:** Okay.

**Stephen Forte:** So I can't say always use this one or always use that one. If you're obviously talking to Oracle and you need something other than one to one table mapping and you want to support Astoria, it looks like something like the Entity Framework or a third party ORM will do the trick if you want to, just, you build something that's a nice application that's going to have a direct access to some SQL server table, one to one mapping, you might want to consider LINQ to SQL.

**Carl Franklin:** This portion of .NET Rocks is brought to you by our good friends at Telerik, without whose support this show would not be possible. If you're a Silverlight or WPF developer, you've heard that having a single code base for your web and Windows user interface is becoming a hot topic. How about building a Silverlight application and then reusing the XAML and the Code Behind for a WPF application? Your customers will enjoy the identical user experience and you will enjoy some free time as you have to write the code for both applications only once. This is not a scenario from the future, the guys from Telerik have developed a line of business demo application that shows you how to do it all. The application uses Telerik's Silverlight and WPF suites which represent two almost identical tool sets for building rich web and desktop applications. Both are derived from the same code base and share a common API, allowing nearly complete code reuse between WPF and Silverlight development. You got to check it out, [telerik.com/salesdashboard](http://telerik.com/salesdashboard) and hey, don't forget to thank them for supporting .NET Rocks!

**Richard Campbell:** Well isn't LINQ to SQL dead?

**Carl Franklin:** Yeah. I mean that's the thing you hear a lot, LINQ to SQL is dead, don't use it.

**Stephen Forte:** And to some degree it's dead, I use this analogy in the election campaign as I said let's just say LINQ to Obama and LINQ to McCain for those of you outside the United States you might have forgotten we actually had a presidential election last year, I bet you're up in Canada, Richard. There's actually was an election, it wasn't just one guy walking on water but what happens is, imagine I said if LINQ to McCain was suddenly told to have had to report to LINQ to Obama, that's kind of what happened inside of Microsoft. The LINQ to SQL Team has now been moved to kind of report and work with the LINQ to Entity Team or in reality is really the Database Access Team. So what happens is, that leads us to all believe that LINQ to SQL is dead and not only does it

lead us to believe that LINQ to SQL is dead. Microsoft is pretty much come out and said it, I mean they've kind of put a better spin on it but they've said it, I've repeated it by saying, by going and putting the LINQ to SQL team to report to the LINQ to Entity Team, in essence they'll kill LINQ to SQL. However, does that mean you shouldn't use it? Well, if you think about it, LINQ to SQL supports a number of features that LINQ to Entity goes not and LINQ to SQL uses the LINQ query syntax. So you could start working with LINQ to SQL and then port your application over to LINQ to Entity or some other ORM in the future, if it was really dead, dead and then what will happen is you haven't really lost much of your investment because the LINQ syntax will be virtually identical and that's the thing that I was showing at my TechEd session is the LINQ syntax between some of these implementations was virtually identical.

**Carl Franklin:** Okay.

**Richard Campbell:** So there's--if I'm a LINQ to SQL guy and I do want to move over to EF, is it painful to move over?

**Stephen Forte:** Actually not that painful to be surprising. Now, if you're going to move over you might not be taking advantage of all of the features of EF. Meaning is you're obviously going to do one to one table mapping to do the port and things like that but the syntax itself is virtually identical. I've noticed some small peculiarities when you do that, meaning if have you have some inappropriate LINQ syntax, meaning your order by is before your where clause...

**Carl Franklin:** Right.

**Stephen Forte:** What will happen is LINQ to SQL will forgive you and LINQ to Entity will not? So then that order by becomes ignored but those are very small minor, we'll call them edge cases but those are some very small minor things, those are the types of things that you have to fix as opposed to syntax. The syntax is virtually identical, not identical but it's virtually identical.

**Richard Campbell:** All right. I don't want to do nothing but ORM discussion here.

**Carl Franklin:** Right.

**Richard Campbell:** Because I think there's more to it than this and you mentioned something casually here about Astoria which I think is incredibly compelling this idea that through an interface like Astoria I can build a Silverlight app that would talk to my database or would talk to Azure -- that to me seems very cool.

**Stephen Forte:** It is actually some very cool stuff, Astoria is one of the kind of coolest things that have come out of the data access debate or the data access, whatever you want to call it, Microsoft's kind of big cook book of database access over the last couple of years.

**Carl Franklin:** Yeah.

**Stephen Forte:** And you can use one programming model, let's just take a step back to what I just said. Richard asked me about, if I'm LINQ to SQL, I want to move to LINQ to Entity framework which should be LINQ to Entities, is it much of a change? I said, not really because your investment in LINQ is really much going to be transferred over. Same thing with, when you think about using Astoria, you're using REST and you're also going to be using the Atom Pub format particularly in REST, so what will happen is, you can then have one RESTful service and learn how to work with RESTful data and even use a LINQ to ADO.NET data services and your client, I'm talking about asynchronously, when you have all this together it is all the same exact programming API. So meaning, as I'm talking to Azure, I'm talking to Oracle, I'm talking to SQL server, I'm talking to as I mentioned before just a collection of memory of ADO objects. If you're using Astoria, as your intermediary between them, you're programming interface on the client is the same, so it truly is a universal data access on that respect. On the back end of course you've got to implement either SQL server specific API's or Oracle specific API's or Azure API's, what have you? But on the client, asynchronously, when you're talking to these RESTful calls, you're really talking to the exact same API no matter what the back end technology is. That's been the holy grail for 15, 20 years and in some respects, at least on the client side, programming side it has been achieved with Astoria.

**Richard Campbell:** Only between those couple of instances but yeah, you're right it's very flexible from there. I just think nobody's celebrating this, it seems like a coo to me.

**Stephen Forte:** I tend to agree, I find it interesting that it hasn't really been kind of front page news, so to speak, but I think what will happen here is, I think to some degree it got drowned out by the LINQ to SQL versus Entity Framework debate that's been going on, well it was going on when Astoria kind of came out because if you remember Astoria was released about a year ago it was released the same exact time as the Entity Framework so we had all of the debates between whether the Entity Framework was a viable product or not, that was one debate and then the second debate we had was LINQ to SQL or Entity Framework and I think missing from that whole discussion just to round out was Astoria and what

people are forgetting is all of the Azure services, so to speak, internally like at Microsoft, like having built these Azure services, they used Astoria, okay, right?

**Richard Campbell:** Ah.

**Stephen Forte:** So you're hitting let's say the Azure table service or the file service like one of those things, anything in Azure that kind of produces a RESTful interface is actually using Astoria and I think people just kind of, it's unsexy building block to some degree, so I think people just kind forgot about it to be honest with you.

**Richard Campbell:** I like it when things stop being sexy because then they stop fooling around, right? It's really just, how do we get the work done?

**Carl Franklin:** Right.

**Richard Campbell:** I want this just to work.

**Stephen Forte:** I'm with you.

**Richard Campbell:** And you keep talking about RESTful and I just don't want to presume that everybody knows what that means per se or why it's good.

**Stephen Forte:** Well, RESTful interface, what REST stands for as Representational State Transfer and when you hear that, you might think it has something to do with the UI...

**Richard Campbell:** Right.

**Stephen Forte:** And that could be anything, so REST is a protocol that is deliberately enforced and wedges itself into the HTTP protocol. So while you have web services that would do things like get and fetch, you would have to learn methods. So here's a quiz for you Richard and Carl...

**Carl Franklin:** Okay.

**Stephen Forte:** If I created a web service, I called it Steve's Hose of Data and you guys will going to call my web service and it's going to produce the statistics for every New York Met baseball player. What would you have to do to get Carlos Mencia's batting average?

**Carl Franklin:** I wouldn't call in the first place because I'm not a Mets fan...

**Stephen Forte:** Okay, I'll also include that evil team up in New England, less evil than the Yankees though, but I would include the Red Sox so you want to get how many home runs that David Ortiz hit, how would you get that?

**Carl Franklin:** It's a web service, not a RESTful service?

**Stephen Forte:** It's a plain old web service, exactly.

**Carl Franklin:** So if it's soap web service you have to get the WSDL and from the WSDL XML, you figure out what the fields are and then you create a soap envelope and then you prepare a, you call a method and you pass the parameters and then you get back a soap message soap result and then you have to parse through all of that and it's kind of hairy, that's why they have those big proxy generators in Visual Studio for you to make it easy for you.

**Stephen Forte:** Okay and you're correct and you said one very important thing, you have to call a method...

**Carl Franklin:** Yes.

**Stephen Forte:** And you would understand what that method was by looking at my WSDL, the Web Service Description Language. So you'd have to call the getplayer method that you might pass at the player ID, you might pass at the, I don't know the other statistics you're looking for.

**Carl Franklin:** Right.

**Stephen Forte:** Well that's how web services work, you expose a bunch of methods so it's action based, right? So it's pretty much remote procedure calls right?

**Carl Franklin:** Right.

**Stephen Forte:** You basically calling a method on my machine, okay? So since I said it, it's set in, you could do those over TCP IP, you could do over HTTP, like SOAP, I'm sorry web services and SOAP goes out of it's way...

**Carl Franklin:** Right.

**Stephen Forte:** To be protocol independent even though we always use it over HTTP.

**Carl Franklin:** You can use SMTP and POP3 if you want.

**Stephen Forte:** Exactly. So REST kind of changes the rules by being more constrictive and by being more constrictive it takes advantage of those constrictions and then it's actually more powerful. So REST only works over HTTP and its resource based not a remote procedure call, not a method. All right so while web services are method and action based,

REST is URI and resource based. So you would use plain old HTTP kind of stateless communication like you get post, put and delete and through addressable resources, I would expose a service that I'll have like steveservice.com/players and you put that, you type that into your browser, you call that with an HTTP get...

**Carl Franklin:** Right.

**Stephen Forte:** And you would get back an XML in Atom Pub or whatever I choose format to it in, most likely something like Atom Pub or RSS or something like that, you would get a list of all the players.

**Carl Franklin:** Right.

**Stephen Forte:** And then you would then put player, then you would say steveservice.com/players/ and you would put that player ID like player ID 5 and that would be Carl Beltran all this stuff, no methods and that's the key here.

**Carl Franklin:** So I'll encode it in the URL all the parameters just like we do standard gets and posts?

**Stephen Forte:** Exactly. So that's why a web service like Flickr is so popular with, has embraced something like a RESTful service because it mimics the unique addressability of an individual photo. So if you go look at photos of Richard and I trekking through Everest last year you can go to like flickr.com/my name/thephotoID and you could do HTTP delete if you had access and you would just delete that photo. You could do an HTTP put and if you had access and you had, you pushed it up with the binary, you can actually post a photo.

**Carl Franklin:** Yup.

**Stephen Forte:** So you're just using standard, so where if I wanted you to delete David Ortiz with a web server from my statistics database...

**Carl Franklin:** Right.

**Stephen Forte:** You would have to call the delete player method or something like that.

**Carl Franklin:** Yeah and you'd also have to authenticated and then you'd have to, all of that stuff which adds complexity and if you want to do transactions, oh my god, now we have more complexity, that's where you need SOAP.

**Stephen Forte:** You are correct because, a RESTful service you still have the necessary authentication, authorization issues. You do not have

transactional integrity in a RESTful service, you can fake it to some degree but you don't have it. So, to some degree it's a bigger debate, web services versus REST as opposed to the original debate we were talking about which was LINQ to SQL to Entity Framework. I would make the argument that REST is fairly generic in so far as that it puts itself into this kind of isolated area meaning you could only use HTTP protocol but by constraining itself to the HTTP protocol, you get a lot of familiar and simple things. Meaning, I can just use my standard HTTP methods like get, put, post, delete.

**Richard Campbell:** It strikes me as very simple, very quick to use, you still can get authentication and some security through SSL...

**Stephen Forte:** But think about it, you can still get authentication, authorization, if you are hosting this, let's just say this is a Microsoft audience, let's say you're hosting this on IIS, it is Astoria and this RESTful service is a, at the end of the day, it's an ASP.NET host so you can put it behind a web config that locks down using form security or using any type of standard ASP.NET security. Of course if you're calling this programmatically, you'll have to pass those credentials in and using encryption, as you just said with SSL but I'll argue that SSL was the second layer of security, meaning your first layer security will be your, the same old application security that's enforced by the web config today and your second layer would be on the web server with things like SSL and even IP blocking. If I want only Richard and Carl to view this RESTful service, I could, in theory, not only standard ASP.NET authentication authorization but I can restrict to just your IP addresses and then force you to an SSL conversation.

**Carl Franklin:** Yeah.

**Stephen Forte:** Because at the end of the day, it works just like a website in that respect.

**Richard Campbell:** And I feel like the whole web service thing with all the WS\* thing just got out of hand like it became unmanageable for regular mortals.

**Stephen Forte:** I tend to agree and I think that's one of the reasons why REST kind of evolved. REST has been around for about 10 years, it came out of a PhD dissertation, I believe the guy's name was Roy Fielding at UCLA, Irvine, I believe about 10 years ago in his doctoral dissertation and looked at it as an academic thing but what happened, the rise of the social networking sites. Flickr was probably one of the first ones to implement a RESTful service and then things like Amazon and then obviously, I said, all the social networking sites like Twitter and Facebook and all those guys, started letting all of their data out as a

RESTful service as well. Much better than having to build some really complex WSDL wf\*.\*, drive me crazy kind of thing.

**Carl Franklin:** In your list, do you include code generators?

**Stephen Forte:** Code generators in so far as what?

**Carl Franklin:** Code generators that generate data access code like ADO.NET code.

**Stephen Forte:** I mean yeah, it's a different conversation, that is not necessarily data access that's just, depends on what the code produces...

**Carl Franklin:** Right.

**Stephen Forte:** If it produces ADO.NET code, I would kind of list that under the ADO.NET category. If it produces LINQ to SQL code or Entity Framework code or NHibernate code, I would put them in those particular categories...

**Carl Franklin:** Okay.

**Stephen Forte:** Or code generation tools.

**Carl Franklin:** All right.

**Richard Campbell:** And there are other protocols that are more REST like, like Atom, even are assessed to some degree right?

**Stephen Forte:** Well actually REST is a protocol, Atom and Atom Pub and RSS are actually all compatible with REST as the actual rendering device.

**Richard Campbell:** Oh, I see.

**Stephen Forte:** So meaning, I can have a RESTful service and it will actually spit back to you Atom Pub or RSS formatted data and that's what the Entity Framework does. The Entity Framework will give you the data in either Json or Atom Pub format.

**Richard Campbell:** Okay and we haven't mentioned Json before but this is just the Javascript notation.

**Stephen Forte:** Right, so Json is a Javascript notation language which is far, far more compact than doing your traditional kind of Ajax call right because Ajax was a term invented by an analyst which I find funny and what Ajax really is, it has the ability to kind of, on the client do Javascript, call web service, get some data and then kind of redraw that page or do something with that information.

**Carl Franklin:** Yeah.

**Stephen Forte:** What we found after the world kind of fell in love with Ajax, it was about four or five years ago now. After the world fell in love with Ajax, for the obvious reasons, Ajax is excellent, we realized that, "Ooh, this was kind of slow." You know because as Carl said, that SOAP envelop becomes so unwieldy and now we're parsing that SOAP envelop on the client and what if the client happens to be an iPhone or something that doesn't have a lot of processing power...

**Carl Franklin:** Right.

**Stephen Forte:** But a lot of people have so that's where Json comes into play because Json is far more compact and now you can do things with Json like `bindputobject` real, real easily, right, because you're basically getting the Javascript, what the Javascript notation language needs, which Json stands for is I need to bind this data to this drop down box so it will give it to you in a nice bindable compact, almost the sheer minimum of kind of text that goes across the wire. So Google started doing it with Google Maps and it really, everyone started kind of following their lead on that but back to your point is if you want your RESTful service to produce Json that's a) perfectly legal and b) pretty darn easy to do. With Astoria it is one little setting when you actually go back and forth with your conversation, the client and the server. You just have your application mime types set into Json instead of XML and Json comes down the wire.

**Carl Franklin:** Yeah, okay.

**Richard Campbell:** Cool. All right, have we missed any data access methods here?

**Stephen Forte:** There are tons and tons of data access methods, right?

**Carl Franklin:** How about data access AntiPatterns, things that we used to do that we shouldn't do anymore or is there any such thing?

**Stephen Forte:** Well I mean it's always best practices and I don't like the term AntiPattern because it makes the assumption that you're right and I'm wrong.

**Richard Campbell:** Right.

**Stephen Forte:** I can do something that is considered an AntiPattern and it may actually work in the scenario I'm in.

**Carl Franklin:** So, it's context sensitive?

**Stephen Forte:** Good example is when I first started coding, I actually had the programming kicked on the main frame and you all heard the terms spaghetti code and main frame and the rule of thumb, what's the first rule of main frame programming? Thou shall not use `gosub` statement, right?

**Carl Franklin:** Yeah, okay.

**Richard Campbell:** Was it a `gosub` or a `goto`?

**Stephen Forte:** `Gosub` and `return` right? So the problem is, in some situations `gosub` was okay, I actually did some work at a bank which is still in business, shall remain nameless, about 15 years ago and I was really writing integration code but I had to dabble in kind of kicks at some point here and there and you actually had to get managerial approval to write a `gosub` statement and if you wanted to write any kind of like modular code, code that we would consider good code today right? The separation of concerns, small method, everything's kind of in a method, that method does one thing and one thing only, if you went and actually did that in Cobol, use a `gosub` statement, that's the only way you really could do it back then. So what's interesting is, I don't necessarily would want to go on record saying, here's an AntiPattern, you should never use it because some, either one of two things, someone is using it and will listen to our advice because then three of us might laugh, "Oh never do X,Y,Z." and the three of us laugh and someone's using it in their application and for some reason we overlooked that scenario and it's working for that user then that user is going to rip it out and start all over or start questioning themselves or b) if I say this is an AntiPattern, I don't want to start some like religious war with somebody else.

**Carl Franklin:** Right.

**Stephen Forte:** I would say that the only AntiPattern is being an absolutist, always doing the same thing, the same way. Let the technology kind of, choose the right technology for the right job I guess in that respect.

**Carl Franklin:** Okay.

**Richard Campbell:** It makes sense to me.

**Carl Franklin:** Yeah, sure does.

**Richard Campbell:** One more technology to throw out there and see how this fits in the context is that the RIA services which you've been really hooked to the Silverlight bandwagon.

**Stephen Forte:** RIA services is actually a pretty amazing little piece of technology and as you said, it's

hooked to the Silverlight bandwagon but I think it's kind of like Astoria was, kind of forgotten about when Entity Framework had its launch, I think RIA services, the fact that RIA services can also hook right into plain old regular ASP.NET is kind of ignored and forgotten about as well with pretty compelling results. For those people who don't know about RIA services, it's an add-on to Silverlight 3, it won't ship with Silverlight 3 it will ship shortly after. So I'm guessing it'll probably RTM probably like early fall, I've no idea but I've been told that it's "after the Silverlight launch" and what it allows you to do is have an object or a service so to speak on your web application and then through a bunch of, I'll just call it Microsoft magic so to speak, there's some kind of code duplication or really the code is kept in sync, there's a lot of proxies and other things that make all this magic happen for you that you can call that service asynchronously from the client and it shields the user from a lot of the plumbing that I think is actually at the moment may be stalling some of the Silverlight adoption and the reason why I say that it's stalling some of the Silverlight adoption is because developers have to really start thinking asynchronously if they're going to be working with Silverlight and as I've said as the original conversation we had once we agree to disagree about clustered indexes is when you're dealing with an application that has a web service and you're in a kind of a Silverlight environment or you're in an Azure environment you're going to have to use an asynchronous kind of methodology.

**Richard Campbell:** Right.

**Stephen Forte:** So even if you're talking to Astoria, your LINQ statements can get really weird because you have to call it and you have to assign a delegate and then you have to catch the event when the delegate's done and then you have to kind of make sure that the event itself is fired and then you fill it all up. I've written a couple helper classes, a lot of people have written other types of helper classes and a lot of developers have said, "Hey, this isn't worth it." I'm just going to stick to plain old, whatever, ASP.NET or something else but I think something like RIA services shields the developer from a lot of that unnecessary complex, it's not really super complex but it's more complex than just opening up a command, executing a data reader and binding it to a grid but it shields the developer from a lot of that and allows them to work asynchronously pretty easily and then work with data back and forth in a nice round trip environment, working validations and all that other stuff.

**Carl Franklin:** Stephen, we're just about down to the end of the show, is there anything else that we've missed that we want to cover?

**Stephen Forte:** I think that there are tons of great resources out there for developers to learn new skills especially data access area. I'd like to call one tool, if you're learning LINQ for the first time...

**Carl Franklin:** Miguel Castro?

**Stephen Forte:** You can use something like LINQpad, it's a great tool to kind of connect to a database and just type in some LINQ queries and have a visualization of the results.

**Carl Franklin:** Okay.

**Richard Campbell:** Where did you get that?

**Stephen Forte:** I get that from [linqpad.net](http://linqpad.net), <http://linqpad.net>, it's free and if you really, really like it and want IntelliSense, you could pay a little extra. It's a great tool, it connects LINQ to objects, LINQ to SQL, Entity Framework and what I like about it is you can even type in regular SQL statements and then, so I could type in a SQL statement, get a result and then I can keep that in one window and then I can start typing some LINQ queries into another Window...

**Richard Campbell:** Cool.

**Stephen Forte:** And start comparing those results.

**Carl Franklin:** Okay, great. Thanks Stephen, it's always a pleasure to talk to you.

**Stephen Forte:** Same here, gentlemen.

**Carl Franklin:** I'll catch you next time at Pepe's in New Haven.

**Stephen Forte:** Sounds great, sounds great to me. We'll get a meatball, onion, anchovy pizza and we'll go into a food coma.

**Carl Franklin:** And only have to wait in line for two hours.

**Stephen Forte:** That's okay.

**Carl Franklin:** That's okay, it's worth it. All right man, thanks 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](http://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](http://www.franklins.net). For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at [www.dotnetrocks.com](http://www.dotnetrocks.com).