



.NET Rocks!

The Internet Audio Talk Show
for .NET Developers

With Carl Franklin **msdn**
and Richard Campbell

<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 #121
(Transcription services provided by [PWOP Productions](#))



Kathleen Dollard Talks Generic
July 8, 2005

Our Sponsors



component developer magazine
<http://www.code-magazine.com>



DATA DYNAMICS
<http://www.datadynamics.com>



Geoff Maciolek: The opinions and viewpoints expressed in the .NET Rocks! are not necessarily those of its sponsors or of Microsoft Corporation, its partners or employees. [.NET Rocks!](http://www.dotnetrocks.com) is a production of franklins.net, which is solely responsible for its content. Franklins.net - "Training Developers to Work Smarter."

(Music)

Geoff Maciolek: Hey, Rock Heads! Stop flaming us for just one second 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 Geoff Maciolek, here to announce show # 121 with guest Kathleen Dollard recorded live, Friday, July 8th, 2005.

.NET Rocks! is brought to you by franklins.net - "Training Developers to Work Smarter". And now offering hands on VB.NET & ASP.NET classes remotely, online at www.franklins.net and by DataDynamics makers of ActiveReports.NET - "Simple Powerful & Cost Effective reporting for Windows Forms and ASP.NET web applications", online at <http://www.datadynamics.com>.

Support is also provided by CoDe Magazine - "The Leading Independent Magazine for .NET Developers", online at www.code-magazine.com.

And now the man who just accidentally inserted the wrong type into his class Carl Franklin.

Carl Franklin: Ouch, I hate it when that happens.

Richard Campbell: That's going to leave a scar.

Carl Franklin: That's going to hurt. So thank you very much. This is Carl Franklin. You are listening to another, the 121st stellar episode of .NET Rocks! - "The Internet Audio Talk Show for .NET Developers". And it's a great horrible drizzly day here in New London, Connecticut on the East Coast of the United States of America. What's it doing over on your side of the North American continent, Richard Campbell?

Richard Campbell: It's a great horrible drizzly day in Vancouver, British Columbia.

Carl Franklin: How does that happen? It's probably beautiful in Texas right now.

Richard Campbell: It's all sunny everywhere else but not here.

Carl Franklin: And usually we don't introduce the guest until after we have had a chance to banter a little bit and read some mail. But since she is right here in the studio I can't really ask her not to

laugh or not to join in. So welcome Kathleen Dollard.

Kathleen Dollard: Hi, how are you doing this evening?

Carl Franklin: I am doing okay. How are you?

Kathleen Dollard: Doing fine.

Carl Franklin: And say hi to Richard out there.

Kathleen Dollard: Yeah, hi Richard!

Richard Campbell: How are you doing Kathleen?

Kathleen Dollard: Yeah, may be it's sunny back home for me in Colorado.

Carl Franklin: Probably is.

Richard Campbell: Oh maybe. But didn't you guys get real snow there?

Kathleen Dollard: Not this time of year.

Richard Campbell: Yeah.

Carl Franklin: Yeah I wouldn't be surprised though. The way the weather is going around here. Well this is like the weekend of Sailfest. And Sailfest is the annual bash that New London throws with tall ships and fireworks. And the Grucci fireworks are supposedly is like one of the biggest displays in the world. And we have a great view of it from the 5th floor here. But it's raining and that puts a damper. I saw my friend in the elevator on the way down and he said "Oh! I got a booth man I am bumming" we can't leave 'em. He has to go to the booth in the rain. Oh well, enough of that.

Richard Campbell: You are not going to believe this. This week is Vancouver Sea Festival and their tall ships are in the harbor right now, and it's raining.

Carl Franklin: You are kidding. This is too weird Richard.

Richard Campbell: It's very spooky.

Carl Franklin: Well, listen to this. Hear that? So not only it is raining but there is a fire somewhere. That's just really weird. Hey at least but we don't have big ocean liners that crash into the docks here in New London.

Richard Campbell: It was a ferry and they have figured out what broke.



Carl Franklin: What, somebody's brain?

Richard Campbell: It was a 75-cent cotter pin. Fell off a nut, nut dropped out released this governor, caused the engines to overrev so they shut off, nice.

Carl Franklin: That's horrible. A 75-cent cotter pin so it's all about...

Richard Campbell: Left a 7000-ton ferryboat without any power. A bow to the captain's credit, nobody died, no injuries. He ran the boat into a wharf like it was a speed bump, destroyed 24 boats along the way. But everybody survived. They've pulled the boat off. In fact it went back into service today.

Carl Franklin: Gees. Talk about your architecture problems huh?

Richard Campbell: Yeah.

Carl Franklin: Well Richard I think it was show 116, where I was reading the mail and it was just another great email of praise and I said "How come nobody ever flames us" and so we asked people to send us flames and what we got was, tepid I mean no flames at best polite suggestions, or criticisms. A lot of them were good criticisms too. They were like "Oh yeah, we could probably do that better."

Richard Campbell: There were some great emails. Great ideas.

Carl Franklin: Great ideas. But I think people are missing the point. We want flames. Because we always get praise and polite suggestions and criticisms. We wanted to get a flame. So we are getting tired of it. So we want some flames. And few people are finally beginning to get the idea. Scott sent us a flame via my blog at weblogs.asp.net/cfranklin. He says, this was great, listen to this, "Hence horrible villain or else spurn thine eyes like balls before me. I'll unhair they head. Thou shall be whipp'd with wire and stewed in brine, smarting in lingering pickle."

Richard Campbell: Yeah.

Carl Franklin: And it's not whipped whipped. Oh no. It's whipp'd, apostrophe d the old English pronunciation. All right well anyway. That was a good one.

Then we get into more traditional modern flames. This one came from Mike Scott in the U.K he says, "Hi Carl and Richard, you asked for flames so here it is - "Your show sucks. I have been listening for about 9 months and I have downloaded about a third of your back catalog to

my pocket PC. In all those shows you have still not expounded a single solution to world poverty or world peace. Okay so Microsoft with a dollar sign say Connect the world together with web services and CLR's the new Esperanto. But who is going to feed the hungry programmers of the world? Who is going to stop the standards wars and hey where is the rock? No Motor Head, no Deep Purple. Hey not even any Pink Floyd. What's with the jingly happy tune at the start of the show? Real rock makes your ears bleed. Why do you insist on making people smarter? My job is all about getting people out of the messes they get themselves into. This making people better programmers is dangerous. If you don't stop it soon I'll be out of a job and then I will have to do some real work. Was that ranty enough for you or do you want some more? Do you still give away crappy stuff. If so please send so that I can burn it in disgust. Grrrr.

That's pretty good. I think we are going to send him a cup or a mug or bunch of boxes.

Richard Campbell: Something that won't burn.

Carl Franklin: Some condoms or something.

This one came from Christopher Brandsma. And he says, "Hello Carl and Richard, all right you arrogant flame loving waste band expanding pats pasty white no sun seeing VB gloating shmucks. You want a flame you can't handle a flame. I read /dot for breakfast and like it. Is that what you want? A vast sea of /dot readers with a collective IQ even smaller than yours. Raving like rabid idiots too, is that what you want? Well apparently if you are willing to appeal to Rory for decent flame, you are. So you Carl you loud mouth blabbering the obvious to your hearts content arrogant singer/song writer, acting like the favorite son of Bill Gates himself, God's gift to binary logic. Never seen a garbage collector you didn't like. Just who do you think you are? Chris Sells?

(Laugh)

But at least I don't have to mentally conjure images of you in a kilt. Or is that yet. Speaking for all the listeners out there. If you have worn a kilt I don't want to hear about it. And what is with Rory leaving? He said he wanted to go. But I don't buy that. My speculation is that you couldn't afford Rory anymore. The show wasn't bringing in the big bucks like you expected, is it? So you had to outsource Rory's job to low lost Canada. That proves one thing crystal clear. You are just a talking head for the man, Carl. Taking another good American job and shipping it over the border. So Richard you scab, look out man, Mexico is right around the corner. Now since I



know your simian brain can't handle all of this, I'll have to resort to subliminal messaging to deliver my final blows. If you social laps (send swag) could talk about some technology (OLap) out side your comfort zone (Analysis services) and get a guest (Moshe pronounced key Roberts here) who could talk, we might have a show (Ralph Kimble somebody please). How about it Carl (VB.NET nazi). Do you dare stepping out (send swag) of that carefully created comfort zone .(I wear an extra large) Maybe then you could grow something other than your perfectly combed hair. (OLap lots of changes in 2005) sincerely Christopher Brandsma. Now that my friends is a flame and Christopher gets a Hoody. That's what I am talking about.

Richard Campbell: Yeah.

Carl Franklin: And so for a formal introduction. Kathleen Dollard is a consultant, author, trainer from NET. How's that for no segue Richard.

Richard Campbell: Yeah

Carl Franklin: All right Kathleen Dollard. She is a consultant, author, Trainer and speaker. She has been a Microsoft MVP since 1998. Wrote Code generation in Microsoft .NET for Apress and is a regular contributor to Visual Studio magazine. She speaks at industry conferences such as VSLive, DevConnection and Microsoft Dev days as well as local user groups. She is the founder and principal of Gen. NET. Her passion is helping programmers be smarter and how they develop by learning to use Visual Studio XML related technologies .NET languages, Code generation, unit testing and other tools to their full capacity. She is currently working on full life cycle improvement such as better de-bugging and capturing business intent in Meta Data and test definitions. When not working she enjoys woodworking, snowshoeing and Kayaking depending on the outdoor temperature. Welcome Kathleen.

Kathleen Dollard: Hi.

Carl Franklin: Wood working huh?

Kathleen Dollard: Not this year.

Carl Franklin: I didn't know about that side of you.

Kathleen Dollard: Yeah, but not this year.

Carl Franklin: Do you get the Bansaws.

Kathleen Dollard: I don't have a Bansaw. I've got a table Saw and lots of tools like that. But the Bansaw just doesn't fit in the garage along with the other junk I've got.

Carl Franklin: So is it like this old house or the new yankee workshop over there.

Kathleen Dollard: No it's more like just playing around a little bit.

Carl Franklin: Birdhouses.

Kathleen Dollard: More like birdhouses, Yeah.

(Laugh)

Richard Campbell: Not making your own furniture?

Kathleen Dollard: Well I take a lot of them in my house until I can get that back up. We got to focus there. And that was 2003's fault. One of the betas took me too long to install and I took a wall down.

(Laugh)

Richard Campbell: Generally it's after I installed the beta and its host my machine that I start taking walls down.

Carl Franklin: Yeah, all right with my head.

Kathleen Dollard: Yeah. But this year I am not doing, either wood working or kayaking. Although hopefully I'm getting back in the shop for the next couple weeks. Because my shoulder is still manning from an injury in January.

Carl Franklin: Oh, yeah that was a mystery injury. We never really found out what happened.

Kathleen Dollard: Ooh! That's right, you didn't.

Carl Franklin: Are we going to?

Kathleen Dollard: Probably not.

Carl Franklin: Is this one of the stupid meter issues.

Kathleen Dollard: Oh, yeah.

Carl Franklin: Okay. Now we got to hear, come on. You can tell us, we are all friends here.

Kathleen Dollard: Okay.



Carl Franklin: I make a bonehead move every year.

Kathleen Dollard: Okay, you got to imagine this. Because it's not funny unless you really get the picture going. Okay?

Carl Franklin: Okay.

Kathleen Dollard: At two o'clock in the morning I am walking across my bedroom. And I catch the toe of one foot in the pajama leg of the other foot. So it's like all of a sudden how I lean forward.

(Laugh)

My feet are tied together.

Richard Campbell: That's awesome.

Kathleen Dollard: Unfortunately I was going into a very heavy under bed dresser that I have with my head and so the good thing about this injury, actually is that I did protect my head. I broke my arm -- I tore my arm up, tore out my shoulder.

Carl Franklin: Really.

Kathleen Dollard: Did a bunch of damage but I am still thinking, which is...

Richard Campbell: Yeah, avoided the concussion but need orthopedic surgery.

Carl Franklin: You dislocated your shoulder, is that what you are saying?

Kathleen Dollard: Well actually, I tore the rotator cuff.

Carl Franklin: Oh, my god.

Kathleen Dollard: And I broke the upper bone. Fractured it in about half dozen places. So we got the bone fixed. And then we did this surgery May 5th. And now I am back to where I --am. I am lifting weights again and so now I am in serious recovery. And so I can see the end of that. But I am not supposed to go back in the kayak this year so.

Carl Franklin: Oh wow.

Kathleen Dollard: It's going to be very-very worse than I thought.

Carl Franklin: That's more serious than I thought.

Kathleen Dollard: Yeah, we are taking one step at a time. And I am happy to be where I am. I can carry my own luggage. So that's a good thing.

Richard Campbell: And you can still write code. That's good. And you can still think. That's the best part as you said.

Kathleen Dollard: Yeah, he never told me I couldn't keyboard. So even when I was doing it where I literally had to pick my hand up to take it from the keyboard to the mouse. My other hand had to lift it. I never had to stop coding.

Carl Franklin: We were down in the coffee shop just an hour ago -- we are going to talk about another ___ if we get all in, in an hour it may take a little bit longer than that. But the last time, you were on the show we were talking about code generation. And you still doing that I guess. But you are getting a lot more into Whidbey these days. And want that's got in stock. Why don't we just touch on code generation a little bit, what you are doing now, if any thing different?

Kathleen Dollard: Yeah, code generation right now a lot of my stuff's been on hold right now. Because I am trying to figure out where to take it for a couple of reasons. One of which is that I think people want a lot more of a full solution. And my goal was to teach people how to do Code Generation. And what was perhaps should have been obvious but wasn't is people don't want to do code generation. They want the end result, which is a fully functional system that they are getting very simply. And they want it very cheaply. And I don't think that those are the wrong goals. I just think that that wasn't the goals I was trying to meet. And so, the stuff that I had out as samples. I think people have tried to take away. And I think there has been some frustration with that. But we've got so much coming in Whidbey that I can't really see -- I am not sure that how much more work I am going to put into 2003 versions. Because I think a lot of the picture changes in Whidbey which is now right around the corner.

Carl Franklin: Yeah I had that same issue with a book that I wrote all this years ago, or the two books that I wrote, is that I got a lot of -- A. Complaints that the DLLs were missing. The DLLs were missing and of course, I wasn't shipping a tool. I was shipping source code as an example of how to teach, what you want to do with it. And people were thinking that it was a product. Oh I am going to get a \$200 free \$400 product for a \$40 book price.

Kathleen Dollard: Yeah, I did have samples that went to in it as a starting point. But signs and



complexities of people's applications vary a lot and I don't consider myself to be – although I am working in that direction quite a bit right now. But especially in the 2003 time frame I really wasn't an architecture guru and that's why I used Rocky stuff because CSLA was there.

Carl Franklin: Rocky Lhotka.

Kathleen Dollard: Exactly. Rocky Lhotka CSLA stuff. Because it gave me a leg up and I didn't have to write about architecture. Anyway that's where we are at. The problem is complexity and complexity and templates it's still complexity and part of my prospective. Because I tend to be more macho than I should be used is that CSLA was extremely challenging template to write at the level that I did. And so they're very complex And so they are very hard to maintain and all those I think are -- it's a whole set of things that I think we can do better. And I am just trying to balance where to go with what we've already got out there. And trying to improve that and I have been in a hiatus because of the injury and all that. from my website to at the same time.

Carl Franklin: And you'll be doing a lot of research into Whidbey.

Kathleen Dollard: Absolutely. I am spending a ton of time in Whidbey. I am actually doing the majority of my consulting work now in Whidbey. I have got a couple of clients that are ready to go there. And there are some awesome things that change the picture as we go forward.

Carl Franklin: So let me ask you this, you've obviously been working with this quite a bit now. What just takes your breath away, what's the coolest thing?

Kathleen Dollard: Well the coolest, it's a little tiny thing.

Carl Franklin: Sure.

Kathleen Dollard: It's Binding List. Binding List of T. Okay, so everything you ever did to make your custom objects work just kind of vanishes into this one little line of code. And all of a sudden you can bind your custom collections to the new data grid view. And boy, it's small. But it just absolutely blows you away how much code that drops out.

Carl Franklin: It's a combination of the generics of T and then a bindable collection base.

Kathleen Dollard: Yes

Carl Franklin: And all that code goes away.

Kathleen Dollard: And the better designs that are in WinForms. WinForms has come a really long way. I think that when we see it we are going to just really feel how infantile the first version was and how much better this version is.

Carl Franklin: Yeah, I totally agree with you there. I'm much more impressed with it. All the little binding details that drove you nuts. Not only binding but of course lack of implantation. I don't know of anybody who uses the real data grid on a WinForms. Most people are using Infragistics or some other third party tool.

Kathleen Dollard: Right and so I think that that's fine. When you need the complexity of those third party tools. But it's really unfortunate to be forced to that just to put a combo box into a data grid.

Carl Franklin: Yeah.

Kathleen Dollard: Which now we can do. So we've moved forward there. And I am easily amused. And so I am really excited about the little line-up things in WinForms.

Carl Franklin: Yeah, that's cool.

Kathleen Dollard: It's just cool. It's really well done and I'm just vastly amused by that.

Carl Franklin: Very cool. So obviously generics is a big deal.

Kathleen Dollard: Yes.

Carl Franklin: We haven't talked about generics. We've given lip service. But we haven't talked about what generics is since Juval Lowy came on and told us quite a while ago now. I don't think anybody was really paying attention to generics. It was like something coming oh yeah it's kind of cool. We don't really know. So what I'd like you to do for the listeners is let's take a big picture approach and step back and spend a little time just discussing what this is and then we'll move on to why people should care.

Kathleen Dollard: Okay, let's start with -- like you say The Big Picture and that's understanding what these things actually are. And what they are is a way that at runtime a new class is created for you. It's specific to the type information that you've have given it. So the type information we can think of is just the class for right now. It also could be structure and interface. But for simplicity let's just think of it as a class. And let's start with the generic method. Because I think that that's a



good entry point into understanding it. So a generic method is a method, for instance it could be a -- my mind just went blank on an idea but it could be any method.

Carl Franklin: SubFoo.

Kathleen Dollard: Thank you, a SubFoo. The SubFoo and -- instead of having the SubFoo just like take an object. Because you sometimes are going to need the SubFoo to do things with integers. And sometimes you are going to need it to do things with strings.

Carl Franklin: By object you means System.Object.

Kathleen Dollard: Exactly. I mean System.Object sounds still to be coming as generic as we can and using all type safety and a lot of performance, we are going to actually create a class at runtime that's going to be of the type we're interested in. And that could be an integer, it could be a string, it could be an invoice class, it could be a serializable attribute class. It could be a little of anything that we want to give that at runtime.

Carl Franklin: And when you instantiate the object you pass the type in? Is that how?

Kathleen Dollard: Yes, you instantiate the object and because that appears in your code then the class is actually created at runtime. And this is if you have done C++ it's very similar to templates. But you want to throw out your concerns about Code blog because they are gone That's not part of the issues and you want to throw out ATL. We are trying to rebuild that, that's the STL of the whole C++. So it makes a monster out of this, that's not what we are doing.

Carl Franklin: Let's assume that the listener has no experience with templates or C++.

Kathleen Dollard: And so once we create this generic method and then we have got it, we can call it as the method itself right now. We don't know anything about this type so we cant do very much with it .So we have do take it one step further in order to useful things and that is to put constraints on it and so for instance if we put a constraint on this that it was going to be something inherited from a Base Biz object that you have in your infrastructure. Then we could do all of the things to that. We could do anything in your Base Bizobject. So we can do those things that are common. Because we have constrained it to that. But the difference is in a very key thing that is hard to get at the beginning is that we are

actually working with that Class. We have a copy, you think of that although it's in memory and so it's not you have to worry about on upload. We have a copy of this method for your Invoice Class and another copy for your Customer Class and another copy for each of the Classes that you have got and so it's very strongly typed. Your compiler is going to tell you if you blow it. So if you create this and you say it's got a constraint of only being something derived from your Base Business Class and you hand an integer, it will blow up at runtime, which it wouldn't if you were taking a system.object.

Carl Franklin: And the thing that I like to think about the Generics and the first question I had for Juval when I first heard about this was, we have variants like in the VB mindset. And we have system object to do late bound things in an inheritance way in .NET. And what's the difference between a generic type and using let's say a variant or an object. The difference to me is that you get the benefit of late binding because you get to decide at runtime what the type is but you also get because it's all built at compile time. You have the type safety and all of the benefits. So it's like having the best of both worlds.

Kathleen Dollard: It absolutely is. And it opens up so many doors to us in terms of where we put our code and that's one of the really exciting things from an architectural point of view.

Carl Franklin: Yeah and so what if some of the, I mean you mentioned the constraints for the business objects. Obviously custom collections become very easily. Your binding list of T is a perfect example.

Kathleen Dollard: Yeah that's great. So what they have done for us is the .NET framework contains a great new set of collections. And I am going to go so far is to certainly say that if you are ever using anything in 2005 out of the old system.collection Class. You should be able to explain why. Now I am not going to say there is no reason to do that because...

Carl Franklin: Backward Compatibility.

Kathleen Dollard: Well Backward Compatibility, Compatibility with backward programmers again these various things you might...

Richard Campbell: It's a different kind of compatibility.

Kathleen Dollard: There may be a reason I haven't seen it yet is my point. I don't know what the reason might be because the new Generic Classes are so compelling. They have a lot more functionality. They have got a filtering they have



got finds. They have got all these new features plus they are going to be strongly typed.

Carl Franklin: Because they can be strongly typed they have these features.

Kathleen Dollard: Some of the things actually I think they could have done before and they didn't. I think they have also have said how do we do collections right. Really right. I think we have new features as well but the combination is really quite sweet. So now if I create a collection and I say this is going to be a collection of invoices. I can't put anything else in there and I can't. Well the only way I can do that in 2003 is to inherit from collection base. And if I inherit from collection base I have a lot of work to do. That work completely goes away and I get it basically for free. And if I have 450 business entities creating something derived from collection base for each one of those. It's going to be a real pain in the backside to do.

Carl Franklin: Yeah. All right so other than the collections, I can hear the questions going on in peoples mind so they are like okay besides the collections when else am I going to use generics? When I am going to build the Class that is in a Collection Class that needs to have this dynamic sort of type feature?

Kathleen Dollard: Well let's look at the infrastructure. And there are a couple of scenarios in which you might want to do this and let's start with one. It's a little bit visual because I think that might help people that are trying to get their head around this. And it is hard to get your head around. Once you see it you see it. But it's hard to get there. So let's say that we are going to create a tree. And this tree was going to have different items at each node. So it's going to have invoices and customers and invoices line items. So this is all going to be in the tree. Now the way that I want to build this is I want to actually attach the objects. It's a fairly small set of data and this helps us see where we are going. I actually want to attach the object to each one of the tree nodes so the tree node is going to have a new item that I am going to add to that. So I'm going to add a new property by deriving from it. Now the way we do that in 2003 is I would recreate 3 new Classes. And those new Classes would be if they will each take one of those things so that could be strongly typed and that's the whole point. I don't want to put just anything in this tree node. I want to put one of these things.

Carl Franklin: You don't want to have to cast it. You don't want to have to return an object and cast it when you use it.

Kathleen Dollard: Exactly, and so now what I can do is I can do that with a generic and I can create a tree now that's going to take a generic. And then if I want to I can put additional constraints and things on that. But then I am explicitly saying okay put in invoice in here put something else in here.

Carl Franklin: It's just a backup so you might have tree.nodes.add and then you are passing in an object and the type of whatever type it is. And so that when you go to access it with the say an item. It's the actual cast to the actual object type.

Kathleen Dollard: Right and in that case I would probably make a Generic Class, which means that the type parameter placed in entire class not just a method. And then I can make my constructor be demanding of that particular type. So that's the part of the way I would approach that problem. And another example is, this is another thing that I am a big fan of Rocky Lhotka. That he has brought out this that when you are creating a new business object if you use the constructor you have the possibility of winding up with an invalid business object hanging around. If you call a shared method then you don't have that possibility because the shared method will either return a valid business object or it's going to return no. Before in order for that we are going to call a create method for right now. In order for the create method to return an invoice or customer or something else very specific we have to put that code into a Class that was specific to that entity. We couldn't put it in the Base Class it wouldn't work. With generics we can make our Base Class can be a Generic Class itself. And once we do that this create method can now move into the Base Class out. And every time we move code out of that business specific entity Class into the Base Class we have done a whole lot in terms of making it a more robust, more maintainable system because we literally have less code to maintain and that should be the goal.

Carl Franklin: Right, the goal is not having to rewrite that plumbing, the collection group for every Class that you need to drag and drop into your project.

Kathleen Dollard: Right, absolutely. And that's beneficial whether or not you are using code generation and it can greatly simplify your templates. And so that's one of the reasons I am excited about generics building architectures that use code generation, use the best possible architectures and use generics all lumped in together. I think that's the real future of where we are going to be and it's really exciting stuff.

Carl Franklin: So I guess the challenge is going to be learning to write code with generics. I mean where anytime that you see something that returns an object you should be thinking how can I make this better.

Richard Campbell: But I think it's even a larger point than that Carl. This is once again getting back to this idea of really maintainable code that it's not above the Classes you knew about at the initial design. It's the classes you didn't know about. The stuff you got to add later is not going to be that difficult to enforce.

Kathleen Dollard: Thank you Richard. That's great. Because it is important to point out that I am now writing code via generics. So I do not know what types are going to run on and do not need to know that. So something can be created later and then it can be used as part of the generics using all of this plumbing infrastructure that we have already built. And so that definitely is an important point.

Carl Franklin: This is so huge yeah.

Richard Campbell: What it is we have had template structures like this before. But we have always had to build them ourselves. I deal an awful lot in transactional processing and we build the set of classes around dealing with – credit card processing because every vendor has got a different formula for how they want theirs to work. And we have always built that framework ourselves and we are really getting it in the box now that generics are going to offer us that sort of base level structures. We don't have to invent this all ourselves.

Kathleen Dollard: That's true and it is going to leave some inventing to do to carry beyond where they have done. The collections they have done beautifully but that's where they have ended it right now. So we still have more exploration to do which can be good. I think we'll learn a lot from it because we will be doing it as a community. And that's one thing I think that's important while it's Patterns and practices group is working hard. It's also important that as a community we develop because what comes out of a collective consciousness I think is often the strongest solution to something.

Carl Franklin: How about Partial Classes? I really think that the Partial Class implementation at least in Windows Forms and in a lot of the Component Class and things that programmers are going to use is really cool because it takes a little bit of intimidation especially for Visual Basic out of the picture. But once you figure out that you can use Partial Classes for anything, there are some very interesting things that you can do

with them and of course there are some issues that you have to think about. So what do you think about this?

Kathleen Dollard: Well again I ask this a lot because people seem to assume that because I do code generation and for certain types of code generation it's great, than I am going to just love Partial Classes and they don't solve all the problems. The way I think people should think about Partial Classes, it's a way to isolate code into a different file. It's that simple. If you try to make it bigger than that I think that's when you get thrown off. One reason we might want to do that is certain types of generated code like with some WinForms. I love when that stuff is gone. I think it's great. So I am really off for that. Well second thing is we may also want to do it for other reasons. For instance if you are working on a project you're under the gun and you have two or three week period, may be not even the life of your project, and you desperately need two people down and dirty on the same Class but in different parts of it. There is no reason you can't split that Class into two files. You can have some muckiness in your source code but it's going to track it. You create a new Partial Class. During that time two people can go as hard as they need to in that code. They cannot run into each other because they're in different files because we have so much in our world that's associated with files, we can take advantage of it on that. But taking a one step further, once we think about that we also to look at where Partial Classes have issues. And the issue is that we worked for forty years developing an object-oriented scheme so that we can have a Base Class implementation and we can overwrite that. That's not supportive with Partial Classes. It is literally one Class. All of those files are slammed together. So I can't have two constructors with the same parameters. These kinds of problems arise. Now there is a way to work with that problem and that is to use events as opposed to using direct methods. Events don't have a basic implementation you are overriding. They have no implementation that you then providing from. So I have a personal dislike of using events for that purpose. And so I anticipate that further code generation designs, I'll be working on Whidbey, I do expect to continue to use the Pattern of a generated Class that then has a derived Class underneath it and it's all of your specialized code. Everything that you're writing, to be very specific goes into that derived Class. And so I expect that model won't go away. Things are as simple as validation, can be pretty tricky to try and do just with Partial Classes in code generated infrastructure environment.

(Music)



Carl Franklin: Folks do yourself a favor and check out our friends DataDynamics' website datadynamics.com - Makers of ActiveReports.NET – Simple, Powerful and Cost Effective Reporting for Windows Forms and ASP.NET. Very nice stuff, you can power the reports right into your application, ship them with your assemblies. Has all the great features you come to expect in a reporting engine and you can use ActiveX controls writing the reports too so, great stuff. DataDynamics has been an excellent sponsor of .NET Rocks! for a long time. They deserve a little bit of your love and attention. So go check them out at www.datadynamics.com.

(Music)

Richard Campbell: I think I might debate with you the idea, Kathleen, that two of us might work on the same Class for an extended period of time. Wouldn't it naturally mean we should break that Class apart? And if it's two completely dissimilar areas in a Class like that aren't we talking about two Classes?

Kathleen Dollard: Well I guess I'm being pragmatic on that Richard. I agree with you but if I am on at a crunch point on a project I'm not going to go splitting Classes.

Richard Campbell: Yeah, that might be more pure but we're not talking about purity here. We're talking about deliverables.

Kathleen Dollard: Yeah, exactly and I'm talking about what do we do when we're running into a problem when two people are smashing into each other and we had no options before, now we do.

Carl Franklin: And that's when you would have two people working on the same Class at the same time because they're both.

Richard Campbell: Yeah in the crisis, the problem is I think what we really done is we're taking known problems and turning them into unknown problems. Okay, we know we have this conflict over of sharing this file. We know we have this Sino Conflicts. And now what we're going to do is split it into two files and we're and introduce a whole raft of new problems we knew nothing about.

Kathleen Dollard: Well I don't know that you're introducing the problems. As long as you perceive it as a single Class it is a single Class. They just now reside in tow files. And as long as you continue to think of it that way then I don't know what kind of problems you might run into. You might. But I'm saying that's actually a pretty clean solution because if I'm going to say it's still

your Class, Richard and you still going to own most of it but everything I start needing to work on I going to start yanking out of there. And that way I'm going to be able to put it some place else and work on it and you can keep it checked out of a source control except when I steal those pieces of code from you and then we have got to work together to get those cut that moved.

Richard Campbell: I'm just thinking stuff like stacking constructors. We're each going to build pieces of this and we are going to run into those kinds of points.

Kathleen Dollard: I'm definitely not looking at when we're first designing the Class. I mean I think then, you're right. If we can identify the problem that early I completely agree with you. I'm talking about crunch point. We're trying to get something done and at that point trying to re-design a Class into two Classes is just not the right answer.

Carl Franklin: Well let me clarify something that you said pretty quickly. So just in case anybody missed it and that is – first of all, when you use Partial Classes, you have a Class split up over two or more files, that's essentially what it is. And then coding those files is put together by the compiler compile time into a single Class, it's great. The example that I want to talk about was type datasets for example because with the type dataset the code is generated now as a Partial Class uses the Partial Keyword. So it'll be easy for you to just create another VB file that uses the same Class, Partial Class structure and then you can add your logic to it, logically speaking this is what you would naturally do, you put your business logic in that separate Class or if you ever re-gen your typed dataset, you're not going to whack that code. And so this is why you said people naturally think code generation Partial Classes and this is an idea actually that Rocky Lhotka brought up on the last show that he was on.

And by the way Rocky is going to be joining us here on the phone in a little bit to talk about some of the stuff a little bit more. He just IM'ed me and said he want to call in.

And in those challenges there as you mention and I just want to slow down and talk about it a bit. So if you've got the genre of the default implementation in a type dataset is to override some of the virtual methods in there and I think one of them is on row changed. On row changed because in type dataset they need to hook that to do something. So if in your second Class here, in your second file of this Partial Class you also want to override that on row change, you can't because it's already overridden in the other file.



So therein lies one of your issues and the thing that Rocky and I were both trying here is what if you grabbed a with events reference to the underlying data table, which you want to get these events on or get these virtual methods on and override and as a different reference to the same object and then grab that somehow in the logic portion of it and then you can handle events, which is what you were talking about. You can handle the events as opposed to the overrides.

Kathleen Dollard: In that case underlying structure in the .NET framework is supporting that by supporting events. And so that's cool. If it wasn't supporting these events and you didn't have control over that Class you would have a little bit bigger problem. And I'd just like to also add that these things are so much one Class but that with events variables that you just add it, if it's already in the other Class the one that you don't have much control over, you can use that's cool, you have got access to it. If you don't you can add it yourself. So you could get the union of all this code and all these variables and everything is literally one Class, which gives you some good flexibility.

Carl Franklin: And a lot of complexity too because you have to know what that code generator is generating. What if your generated code needs to call out to your logic code? You really shouldn't do that and that's where you run into-- bump up against all these design issues.

Kathleen Dollard: Right. And you're also going to have a system and especially with the typed dataset, I'm not sure overtime exactly what this is going to mean but you will be more fragile to changes in the template that creates, which in the case of Microsoft it's big gob of code. But if Microsoft changes what a Strongly Typed DataSet looks like. They'll be more likely to break your code with the Partial Class than they would with the derived Class, which is also supported. As I understand it with the Strongly Type DataSet. I actually having a chance to play with them a whole lot.

Carl Franklin: And speaking of all of this stuff our friend Rocky Lhotka is joining us on the phone. Where are you Rocky?

Rocky Lhotka: Today I am in Sunny Minnesota.

Carl Franklin: Oh okay. Back at home?

Rocky Lhotka: Yes.

Kathleen Dollard: You found the sun. Earlier we couldn't figure out where it was sunny -- sunny in Minnesota.

Carl Franklin: I knew it was sunny in the middle of the continent.

Rocky Lhotka: It's beautiful here. Couldn't ask for better.

Carl Franklin: It's raining on both coasts apparently. Well you've been listening hanging out in the chat room, have you?

Rocky Lhotka: I have. I've been listening all along here. It's a very interesting conversation.

Carl Franklin: So I am sure you would have lots of things you'd like to talk about here. Points you'd like to make.

Rocky Lhotka: Well I've got a short list anyway of things that I thought were interesting. First off, I suppose that the idea of the CSLA Templates are complex. It's probably true. I think Kathleen did a nice thing with the showing people how to do code generation. And I think the reality is that any complex business system ends up with a lot of code. And being able to automate some sort of generation or some of that at least is really a powerful thing.

Carl Franklin: Seems like the two of you are made for each other. Code gen and CSLA.

Kathleen Dollard: But we also argue a lot.

Carl Franklin: So Rocky you've been doing this. We were actually talking on the phone on IM rather about this Typed DataSet code partial class dilemma. What've you learned?

Rocky Lhotka: It's been kind of a mixed bag. I've been spending actually quite a bit of time on this over the past, well since Tech Ed really. And it's exciting and thrilling and at the same time it doesn't quite work. And I'm hoping that I'm running into problems with Beta 2.0 and the things will be resolved before release.

Carl Franklin: So what are some of the issues?

Rocky Lhotka: Well specifically what I'm trying to do is take this idea of partial classes attached to a data table and really use them as in build an application that a person will be reasonably happy with in terms of editing data and viewing data and adding and removing data. And my goal in what I'm doing is to put all of the code, all of the validation and any calculations and anything else like that into the data table. Basically trying to make it become a smart data container. And you can do that to a reasonable degree because you get a column changing and a column changed event. And exactly what you guys are



talking about or you can't override but you can respond to an event.

Carl Franklin: We're talking about real time validation here, what we're talking about, right?

Rocky Lhotka: Yeah. Essentially as the user tabs off field or moves from one cell to another in a grid, the data bindings put the value directly into the data table and that triggers the column-changing event.

Carl Franklin: Right.

Rocky Lhotka: And in my case then that triggers any validation or other business logics that I want to have occur at that point. And that works really quite well except like I said I have run into a couple of quirks. For instance the data grid allows the user to press escape or there's a variety of other ways in which the user can trigger a roll back of any changes in the row. And so if you've been editing the row and say that you've got an invalid value in one cell automatically error icon will show up which is very nice. And then you hit the escape key and the value gets reset to its original value but the error icon stays. And the reason it does that is because in the case of a rollback no events fire in the Partial Class. So I was just smiling listening to what Kathleen was talking about because this is exactly the challenge. Partial Classes are awesome but only if the generated code raises the right event. If the generated code doesn't raise the right events at the right times, they're useless. I mean there is no way to work around or get in the middle of what's going on.

Kathleen Dollard: Yeah. And I'm afraid that that touches back on one of my real core principles and that is Microsoft hasn't bought into yet. I hope they will soon, which is that you've got to be able to control that generated code and we can't in the Strong typed DataSet. And it really reduces it's value because if we run into a wall, the wall is so big that you have to reverse engineer, we use a different word of course, the Strong Typed DataSet into templates yourself. And if you're doing that why were you using Strong Typed DataSets in the first place. So we need them to work really well if they're going to force us to use exactly their code.

Carl Franklin: However well hidden there should be a template somewhere.

Kathleen Dollard: Yeah. And I hope they'll be ready for that in Orcas because there are about sixty different places they could do that and improve the framework greatly. But it's a big step for them and it's going to take a while to get there.

Carl Franklin: Well also it could be, they have to make it known that if you change the template we don't support it. And how are they going to know if you've changed it?

Kathleen Dollard: Oh I think it be easy for them to know.

(Laugh)

Carl Franklin: I mean when somebody calls for support the template isn't working. I think maybe more often than we realize I think Microsoft is very concerned about support.

Kathleen Dollard: I don't buy that actually. I do buy that they're worried about support. I don't buy that on the Strong typed DataSet that that's more of an issue than today because today how do they know you didn't go in and change the code.

Carl Franklin: Well that's true.

Kathleen Dollard: So I think we can do some check sums and things to solve that problem if it becomes when they really want to solve. But at this point I think it's the technology jump is so big because they'll do it at a very high level if they do it. And I don't know that we're all ready for that yet. And so I think there's just a lot of work. And that they were doing other really good things on Whidbey and they just didn't make it. So it's interesting. And they've improved the Strong typed DataSet too. So you can do more with it. It's at least the ones that I saw, I haven't worked with it recently. But Rocky have you looked at deriving from the strong typed data set and perhaps even trying to get a hold of some of this?

Carl Franklin: I've tried that myself. Have you tried that Rocky?

Rocky Lhotka: I have not. Basically I'm cynical like most of us. But I thought, hey let's just give Microsoft the benefit of the doubt and assume that all of the marketing stuff about all the new data binding. Let's just assume that it's as good as it sounds then see how far it can get. And thus far I haven't gotten very far.

Kathleen Dollard: Well on the good side also Rocky though, if you wind up with this not being something that pans out real well. It is so much easier to bind our custom collections and custom objects than it was before.

Rocky Lhotka: Oh Absolutely true. And my real passion is absolutely still in the object oriented base in CSLA and all that. In a sense I wouldn't say it's a lark but it's definitely a tangent for me.



Kathleen Dollard: Hey we have a good word for that now, did you know? We get to steal a word from extreme programming and now we can call it a 'spike.' If we call it spike we get paid for it.

Rocky Lhotka: This is a spike. Excellent.

(Laugh)

Carl Franklin: I think I did try this what you have suggested Kathleen, which was using a type data set as a base class. And I didn't get very far because I found that I was in casting hell. And I can't remember exactly what the implementations were that made me shake my head and say, nah this is stupid.

Kathleen Dollard: Was that 2003 or 2005?

Carl Franklin: 2003.

Kathleen Dollard: 2003. Forget it. Don't try it. Don't do this at home with children. It won't work.

Carl Franklin: But I haven't tried it in 2005. And you're saying there's some more support for this now or...

Kathleen Dollard: I saw and like I said I haven't looked what's in Beta 2.0. I saw some early stuff that they were going to work on this problem that they understood that is bad enough that we couldn't get to the generated code it may be useless if we also couldn't derive from it. And so they certainly had that on their radar. They did work on it. I haven't looked at it I'm sorry. I recently saw it I just don't know what's in there. Rocky have you looked at deriving from the new Strong typed DataSet?

Rocky Lhotka: No I haven't.

Kathleen Dollard: Okay.

Carl Franklin: Well there is some homework for us for the next time we get together. And any of the listeners who want to dabble around and let us know what they find. That's a good thing to do. Well there you go. That'll bring this conversation to its knees, don't you think?

Kathleen Dollard: Yeah when we all don't know something.

(Laugh)

Carl Franklin: Now what'll we do?

Richard Campbell: Now we got to stop and go work on some code.

Carl Franklin: Well Rocky is there anything else that you want to tell us that you're interested in these days that fits in with the topic we're talking about?

Rocky Lhotka: Well I guess I've been playing and I think a lot of people have been playing with all of the same things that you're talking about in terms of working with generics and finding out their strengths and weaknesses and like Kathleen noted you can use constraints on them. But you can never use operators because there's no interface that defines the + operator or base type. So it turns out that they are as an entire class of Generic or what would have been called templates in C++ that you just can't do. And I spent sometime trying to figure that out because I have worked at a couple of C# workarounds for this where they tried to define in interface that had every operator. So it doesn't actually work because you can't grasp that interface onto an integer. And interestingly enough VB has got an interesting answer whether it's good or bad I guess we are up to the listener but late binding solves the problem.

Kathleen Dollard: And a lot of these problems, Rocky related to the primitive types. When we are going to working with something like the business object like an invoice or customer. We are not too likely to be using operators on the object itself. So a lot of that's related to...

Rocky Lhotka: One would think not. I got to say that while it is nice to have operator overloading, it's not all that useful in day to day life because that's you don't put it on a customer or in invoice and so doesn't impact this is much as you would expect, right.

Kathleen Dollard: Right. And there is couple of other things on Generics while we're on that. There are things people might run into and it's nice to know they can't go there. One of which is if you create a new object you do have to have your type with the constraint of new and that has to be parameterless constructor does not support parameterized constructors. And there is also class of things that, if you get an operator part at least would want to do, when we said this is constraint to a numeric. I don't care if it's a floating point or decimal or integer but it needs to be numeric. Well we can't do that and there are a number of things there that we would like to see work better and Generics will better as they grow up.

Richard Campbell: You want Generic constraints now as well.

Carl Franklin: We have those.



Kathleen Dollard: Well we have Generic constraints. We just kind of don't have constraints on the constraints.

Carl Franklin: Well Rocky, listen, it was great to have you call in the show and I look forward to talking to you some more after or around ship time after we have had chance to really dig in into these thing and we are all looking forward to CSLA.NET 2.0.

Kathleen Dollard: Oh yeah.

Rocky Lhotka: Yeah all right well thank you. Really this was fun.

Carl Franklin: All right cool thanks.

Kathleen Dollard: Bye, bye.

Rocky Lhotka: Bye.

Carl Franklin: So Kathleen, I know that you have a stupid meter that things register on once a while.

Kathleen Dollard: I think we all do. I think we all have a stupid meter and I think programmers are not paying enough attention to it. So let me tell you what I mean by that because I don't want anybody to think I am saying you are stupid and I just like fall over my Pajamas and I admitted that. So that's not what I am talking about really with the stupid meter.

(Laugh)

Well what I am talking about here is that we come up with things and there sometimes things are like the world is like telling us. There is something like extra-moral if we do these things or something. And we are like rebelling and we just like refuse to do it and I think that there is some really big ones that are out there. And that if we recognize this we can both have Microsoft make a better product and we can understand why we're sometimes not doing things we think we should. It's like for instance I talked about tracing and I talked in a room and I will say, okay who traces today and people look so guilty. I mean I wish I had photographs. They are like, we know we should, no you shouldn't. Tracing stinks in every language that I have ever seen up until 2005. It stinks -- it's fancy PrintF. I mean okay yeah we've done some extra stuff with listeners in 2003.

Carl Franklin: And tracing just for people who don't know what it is, it's basically a way to output data in your program to a log or some other kind of log file or some other device. So that you can look and see what happened in your code and

when and you can have some diagnostic tools. That's what it's for.

Kathleen Dollard: Yes.

Richard Campbell: Well then we used to just do with printFs now they're trying to give us tools that do the prinoffs automatically.

Carl Franklin: Debug print, yeah.

Kathleen Dollard: I go into the 2003 and I do a right line statement. So trace.rightline and I'll say okay what should we put here and we get stuff like OOPS. And we are likes as OOPS 42. And that's my favorite one.

(Laugh)

Okay so if you think about it, what you are saying is. I was at this point in the code. That's what you're trying to say, okay I was here. And so what do you actually do with that as you actually get this big file with all this that you have in it you loaded into Notepad of all things if it's too big to put in WordPad. Gosh that's the step up and you use the find feature to find OOPS 42 to see whether or not it happened and when it may have happened. That's craziness, it's absolute craziness. If you use a Notepad your stupid meter should already be pegged.

(Laugh)

Carl Franklin: I hate to tell you this but I use grape for that so I can tell you of had 17 OOPS 42s.

Kathleen Dollard: Okay so we have move forward here. We got from Notepad to Grape. This is not constructive, it's not we need to be with tracing because it's actually a very, very valuable feature. If you could actually say I was here, I was there and you could put lines in your code to under control here and you can stick stuff in there about what was going on that would be extremely valuable during debugging and so it's a good thing. It just doesn't work in 2003. And in 2005, we fix that but...

Carl Franklin: When you say it doesn't work I mean...

Kathleen Dollard: It works.

Carl Franklin: It works. It just doesn't do what it should do.

Kathleen Dollard: It doesn't work very well in the real world. There is not a way to get consistent output. There is not to give the control you need to the consumers. And really that's one of things



by tracing is that tracing is about the consumer. You may be the consumer a madman programmer in 5 years may be a consumer, a sys admin may be the consumer of this information. And so different then most of the stuff we do within our code we have different set of consumers to think about. So the other one I love on stupid meter is test-driven development. Let's all sit around and get drunk because we are so guilty that we are not doing test-driven development.

(Laugh)

Hello I am doing test-driven development and I swear I wish this wasn't true because I really like test-driven development. But the first piece I did I was actually doing some stuff where I was enabling buttons and it was silly little job. So the job in application was pretty easy. So I wrote up some infrastructure for the test-driven development and I did all this work and I got my test ready then I retried the code. So the code is like 25 or 30 lines. So I am doing matrix on this project because I want to know where we're at. I swear the lines of code in the test project that's side of it, 666. So we started out with the market of devil. Really I swear that's what it really was.

Richard Campbell: Take the hint.

Carl Franklin: I am the god of hell fire.

Kathleen Dollard: I got down to my incremental. So every time I'd write a new line of application code I was only writing 1.6 new lines of test code but I never got it down to the same. This is for very simple problem so a more complex problem it might not be quite as much. And I am not seeing anybody actually doing test-driven development. I mean I am saying people trying but it is too hard and we have got to make it easier.

Carl Franklin: Such a change. It's such a change in the way you write code.

Kathleen Dollard: It's not just a change in the way you're writing code. You are adding perhaps 4 times as much work. You are adding say 3 to 5 times as much code, may be twice as much code to the application and you are going to your manager and saying I am going to work whole lot harder and I am going to make better application. I believe in test-driven development. But we have to bring it down in order of magnitude of 2. And this is something that's going to require a lot of thought by a lot of smart people. I do think that there is some ways to do that. I am working on something called declarative unit testing and what that is certain types of tests we know about. Okay if we have a parameter that shouldn't be,

'No.' Okay we can say we already have XML as comments, which I think we should stop calling XML comments and the funny thing here is that I fall like had to keep XML comments out of VB and now they are there and I am using it but it is...

Carl Franklin: Why?

Kathleen Dollard: I don't want them.

Carl Franklin: Why?

Kathleen Dollard: Because angle bracket summary, angle bracket a bunch of gunk...

Richard Campbell: **Carl Franklin:** It's 2 or more of us it could be anything.

Kathleen Dollard: No, no there is no value over summary colon. It's a bunch of extra gunk that it makes hard to read for no value. And anyway but now it's there and if we just think of it as XML Metadata function and that Metadata can be used for whatever we want it to be. And one of those things that we want it to be is documentation.

Carl Franklin: And Intellisense yeah.

Kathleen Dollard: And Intellisense. We want those things but we can simply add a new attribute to the parameter tags it's already in there you can do this in 2003 you should be able, I've got testing in 2003. Now we can build a test for that no, which is the test we're skipping today actually in test driven development. It's a very important type of testing we can have. We should be raising the standard exception. We can build the test. We can do all this automatically that's actually work. I wish how much people start working on it because I don't have time to do that I mean it's just too much.

Carl Franklin: You know what else this -- the issue with the test-driven development it's if you're like me and you architect and write at the same time. You can't really do with test-driven development. You have to know what's you are going to -- you can't just sort of experiment around that's the way you can with.

Kathleen Dollard: There is an extreme programming answer to that. I think it's important to share whether or not, it's reality or not and that is what the word spiking actually does mean. I made a joke about it earlier but that says that we will have time we're writing real application code using all the rules including test-driven development. We will have other time when we need to experiment. We need to dive and we're diving off the diving board into the deep end and



that's spiking and it is important and my spiking code is going to wind up in my application. I mean I am just not it's going to happen.

Carl Franklin: Sure. Well okay now I ask everybody, who does my show these days. What's the coolest thing you have downloaded lately?

Kathleen Dollard: Well I am going to give a plug here because I actually do thing this is the coolest thing I downloaded lately. But if you go to clarkdollar.com, you will find some music and this is Clark and Clark ... depending on how many tracks you use for the particular song. And it's music that is combining some of his life philosophy, the zen based type stuff and music.

Carl Franklin: Who is this?

Kathleen Dollard: He is my brother. So it's a pretty cool stuff I think.

Carl Franklin: So if we can play a clip from one of the songs here, which one should we do?

Kathleen Dollard: Oh! Gosh. It's one of them called "Lawnmower." Pick any if there is not the word "lawnmower."

Carl Franklin: There is "Thank you for breaking my heart."

Kathleen Dollard: Oh that's a cool one.

Carl Franklin: Songs for the refrigerator door.

Kathleen Dollard: 'Thank you for breaking my heart' something for that that's the new one. 'Thank you for breaking my heart' is a good song.

Carl Franklin: Okay this is an album.

Kathleen Dollard: Yeah it starts out with the albums and then it goes to...

Carl Franklin: So I should do the first one here?

Kathleen Dollard: Yeah.

Carl Franklin: All right let's just listen to little bit of it. Do you think he'll mind?

Kathleen Dollard: No I don't think he'll mind at all. And feel free to download the music if you want to.

Carl Franklin: All right hang out while it downloads here. Little plug for Clark. All right here we go.

(Music)

Carl Franklin: Hey that's pretty good, very cool.

Kathleen Dollard: I think so I told you it's cool.

Carl Franklin: That's cool.

Richard Campbell: And it's a something a semi hollow body type of guitar. He still play guitar. It's got a nice sound to it.

Kathleen Dollard: Yeah he is got a couple of really sweet guitars that he uses on the thing and he plays on base and things and so that's a lot of interesting stuff on there.

Carl Franklin: Awesome, So where are you headed to next.

Kathleen Dollard: Well, Monday night I am at the Vermont User Group, which should be the night that this comes out and then I am at Montreal on Wednesday. I am in Los Angeles August 1st. And then in September I am doing a trip with, September and October I am in Charleston, Spartanburg, the Raleigh-Durham Area. And one more then I might be doing Texas A&M as well. So I am doing a couple of trips, where I am getting out and doing a couple of things. And I am really enjoying the users groups and talking and that's a lot of fun.

Carl Franklin: Okay are there any good resources you can recommend for people who are getting into the stuff -- Generics, code gen, Partial Classes all that stuff.

Kathleen Dollard: Well tracing is really good when it's for a reason. And John Robbins just puts something, it's an MSDN magazine, which means it should also be on the Internet and MSDN. And it's his column and it's the last month of the month before. I am sorry that I don't remember but I think tracing is in the title. He is does a pretty good job of covering tracing.

Carl Franklin: I think it was in Bugslayer wasn't it?

Kathleen Dollard: It's in his Bugslayer column, yes. And I just started remember which Bugslayer column it's in.

There are a couple of things that I would take further in terms of importance but basically he covers the basics and starts giving you a view of it. But he doesn't understand why we need delimiters. We need to delimiters, he says like the delimiter output is like why would we need that if you like to use it. You like it because then you get upgrade from Notepad to Excel and that's a



really, really big step because you can do sorting and filtering.

(Laugh)

Carl Franklin: Okay. All right, cool. And anything else you want to say, last minutes words of wisdom before we say good-bye here.

Kathleen Dollard: Get ready for 2005. It's absolutely huge, it's a very, very good. It's a very positive, huge steps forward but it's big. It's not a baby step you are not going to get there overnight. The good thing is your code will but you won't; you're going to have a lot of work to do, and enjoy because it's going to be a lot of fun.

Carl Franklin: Awesome. Well Kathleen it was great to have you in the studio.

Kathleen Dollard: It was great fun.

Carl Franklin: All right well on behalf of myself, Geoff Maciolek in sound room, Richard Campbell in Vancouver thanks for listening .NET Rocks! You guys have a good week and keep those flames coming, we need a laugh. All right, thanks again.

(Music)