



<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 # 232
(Transcription services provided by [PWOP Productions](#))



Jeff Atwood on the Human Side of Software Development

April 26, 2007

Our Sponsors

CoDe
component developer magazine

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

 **telerik**

deliver more than expected

<http://www.telerik.com/>



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 FranklinsNet, which is solely responsible for its content. FranklinsNet - 'Training Developers to Work Smarter.'

(Music)

Lawrence Ryan: Hey, Rock heads! Put down that book on subliminal persuasion 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 #232 with guest Jeff Atwood, recorded live Thursday, April 19, 2007. .NET Rocks! is brought to you by FranklinsNet - 'Training Developers to Work Smarter', and now bringing the Just-In-Time Team System Class with Joel Semeniuk onsite for your development team online at www.franklins.net. Support is also provided by 'Telerik' - combining the best in Windows Forms and ASP.NET controls with first-class customer service, online at www.telerik.com; and by 'CoDe Magazine' - the leading independent magazine for .NET developers, online at www.code-magazine.com. And now, the pwoppin' man, responsible for every pwoppin' pwop in the pwoppin' show today, Carl Franklin.

Carl Franklin: Thank you, thank you very much and welcome back to .NET Rocks! It's Thursday, this is the Thursday show. I am Carl Franklin, here in New London, Connecticut at Pwop Studios, Pwop Central and Richard on Pwop Campus West, out there in Vancouver, British Colombia. Richard Campbell, how are you?

Richard Campbell: Here I am on the left coast, making shows as fast I can. RunAs Radio, going like crazy.

Carl Franklin: Yeah, we are at show #3, this week. Yesterday's show was awesome. Just good stuff.

Richard Campbell: Yeah, Dana Epp, who's a local guy, I have met him before, he is here in the Lower Mainland. We just had a really good time. I think, it came across to the show, we were laughing most of the time.

Carl Franklin: And it's good. It's good to get that IT perspective as a developer. We've done a lot of IT focus shows but you guys really take it to a new level.

Richard Campbell: Well I'm talking about Card Space too which we already did, we talked to Kim

Cameron on .NET Rocks! about Card Space, from a programmer's point of view, totally different context to talk about from an IT point of view, what Card Space means?

Carl Franklin: Absolutely, all right well, speaking of stuff that we talk about every week, we talk about emails, we read emails, that is and here is the one that came in from Bill Ayers, the subject is the .NET Rocks! fitness plan. Oh, boy, here we go. "Hi, Carl and Richard, I have been listening to .NET Rocks! for a very long time, so, no need to suck up"

Richard Campbell: Nice!

Carl Franklin: "A couple of things, number one, in show number 225 with Dan Appleman, there was a discussion of early Micro Computer systems and you ask listeners to beat the system, he described which only had 16K of memory. The first computer I owned, I designed and built myself using a National Semiconductor SC/MP Microprocessor. The S stands for Simple, and it had a tiny instruction set by modern standards, in fact it was possible to memorize all the Op codes and program in machine code without reference to the manual. I had two 256x4 bit static RAM Chips, that's right kids, 256 bytes of memory, not 256 kilobytes or megabytes, just 256 bytes. Can you beat that? I can. At school I used a computer built as a project which used discrete transistor switches and filament bulbs and had 16x8 bit bytes of memory".

Richard Campbell: Brilliant, that would have taken up a lot of room too, because it was just all discrete hardware, so each one of the bits would have probably taken up the better part of a square inch.

Carl Franklin: Oh, my good that's insane. Okay, "uncle!"

Richard Campbell: Uncle.

(00:04:58)

Carl Franklin: All right, "Number two, you keep reading out letters about this or that person, who listens to DNR while running, cycling, swimming the Atlantic, climbing Mount Everest, etcetera. After reading the letter you say, that's amazing blah...blah...blah... we are sending you a container load of DNR swag because your letter is so, original. I would just like to point out that, I was the first to draw attention to the .NET Rocks! fitness program, way back in show #120, with Sanjay Parthasarathy, where you kindly read my letter and from which I scored zero swag. Two shows a week is great news. I am adjusting my training schedule accordingly." Best regards, Bill,



Dr. Bill Ayers, from Flow Simulation Ltd. in Sheffield England, that's flowsim.com. Give 'ya some props there doc. And yes, I am sorry. You know, we don't have a swag brew program here, you know. Sometimes it slips our mind and I am sorry.

Richard Campbell: Obviously, he needs a Hoodie,

Carl Franklin: I am going to send you a container full of swag, right, we are going to send you a palette.

Richard Campbell: A palette load.

Carl Franklin: Dude, you are going to have to build a new room for the swag that we are going to send you. Alright, you have out-geeked us and you have out-classed us.

Richard Campbell: Totally.

Carl Franklin: Congratulations.

Richard Campbell: I got an email as well, although, not quite as intense, a little more technical. It says, "Hi guys, I was just listening to your show with Oren Eini, great stuff as always. I have used Rhino Mocks a lot and I quite like it, for what it achieves. However, I am of the opinion that we shouldn't need Rhino Mocks or Type Mock or whatever at all. I think, Unit Testing should be a platform service that is, the CLR itself has intrinsic support for unit testing and all that it entails. All of these unit testing frameworks have limitations that come about by the lack of platform support for unit testing. I have found these limitations extremely frustrating and I often have to spend too much time unit testing a piece of code, or I have to skip unit testing, of that code, altogether. Suppose, unit testing was a platform service, it might be possible to run a CLR in a special unit test mode, which accompanies all of the necessities for unit testing. Unrestricted comprehensive automatic mocking, unrestricted access to the code under test from the unit test code, access to private members for example and so, on. Obviously, a lot of thought would have to be put into this, if it was ever going to be implemented but I just wanted to hear your thoughts on this, kind regards, Kent Boogaart." Kent you blew my mind.

Carl Franklin: You want to hear my thoughts, I think I need a nap. That's what I...

Richard Campbell: This is something, I have been thinking about in the sense of, it seems silly to me that the CLR, that's loaded on the client machine is the same CLR that we are running on our Dev Machines, obviously they need to be

similar but there really ought to be a debugged version. I mean what he is talking about makes sense in the sense that, there ought to be an instrumentation harness over top of the CLR.

Carl Franklin: There should be.

Richard Campbell: So, that we know more as developers.

Carl Franklin: And you know, rightly so, it should be standardized, I mean, to have it come in the box, would be really wonderful.

Richard Campbell: Just to guarantee sort of minimum standards for everything.

Carl Franklin: Right.

Richard Campbell: So, interesting thought Kent, we can't do anything about it.

Carl Franklin: Wish I could.

Richard Campbell: But we will pass it along to Microsoft, how about that? They can ignore us, instead of you.

Carl Franklin: And I think, we just did.

Richard Campbell: That's true.

Carl Franklin: Let's get on our announcements, the conferences DevTeach and TechEd. DevTeach, up in Montreal, of course devteach.com.

Richard Campbell: And that's May 14th to the 18th

Carl Franklin: Yeah, and beautiful place, beautiful conference very small, very intimate, nice way to pow wow with the talent.

Richard Campbell: And we'll be there.

Carl Franklin: Yeah.

Richard Campbell: And we are doing the ORM Smackdown.

Carl Franklin: The ORM Smackdown, should be a lot of fun, also TechEd June 4th through 8th we are going to be all over that conference, walking around...

Richard Campbell: Indeed.

Carl Franklin: Doing stuff and I think, we are going to do the quiz show there, right?



Richard Campbell: Yeah, we are looking at the quiz show, talking about some kind of variation on speaker idols still working out the details on that and of course, we are going to be recording lots of shows from the floor and we are discussing a panel as well. That will be a show and that's June 4th to 8th Orlando, your favorite place in the world.

Carl Franklin: Okay, get your pencil, here comes the Code Camps. The Twin Cities Code Camp, Twin Cities, Minneapolis and Saint Paul, of course. Rocky Lhotka is going to be there, you can guarantee that. April 28th, you can read about it at shrinkster.com/o9d.

Richard Campbell: There is also the Calgary Code Camp, up here in the Great White North, also April 28th that's this weekend, at shrinkster.com/o9f.

Carl Franklin: And we are talking about the Ann Arbor Day of .NET, Ann Arbor Michigan, May 5th at shrinkster.com/cuk.

Richard Campbell: And also May 5th is the Austin Code Camp in Austin, Texas, one of my favorite places.

Carl Franklin: Great music.

Richard Campbell: shrinkster.com/o9e.

Carl Franklin: And that's also May 5th. The West Michigan Day of .NET, May 19th, shrinkster.com/nih

Richard Campbell: And finally, June 23rd the Raleigh Code Camp at shrinkster.com/o17.

(00:10:03)

Carl Franklin: Raleigh North Carolina, June 23rd and of course the .NET New York tour, is still going on, Greg Brill is looking for you, a talented .NET developer, if you want to spend a year in New York City and live rent free in a great apartment down there and work in an exciting industry and company go to shrinkster.com/kh6 to learn about that. Also if you are a hotshot ASP.NET guru, there is a gig for you in Washington DC, if you're located near or willing to be relocated to DC, go to shrinkster.com/mmj.

Well, Richard this is a very special show because our guest today is Jeff Atwood, who is the head blogger at Coding Horror, which is his blog. And I am just going to read his bio because it's in the first person. So, this is about me, "Who are you?" "I am Jeff Atwood. I live in Berkeley California, with my wife, two cats and a whole lot of computers. I was weaned as a software

developer on various implementations of Microsoft's BASIC in the 80's .starting with my first microcomputer, the Texas instruments TI-994a. I continued on the PC with Visual Basic 3.0 and Windows 3.1 in the early 90's, although I spent significant time writing Pascal code in the first versions of Delphi. I am now quite comfortable in VB.NET or C#, despite the evils of case sensitivity. I consider myself a reasonably experienced Windows software developer with a particular interest in the human side of software development, as represented in my recommended developer reading list. Computers are fascinating machines, but they're mostly a reflection of the people using them. In the art of software development, studying code isn't enough; you have to study the people behind the software, too.

I currently work for Vertigo Software in Point Richmond, California. Vertigo is gracious enough to host this blog. You can take a virtual tour of my office if you'd like."

Welcome, Jeff Atwood!

Jeff Atwood: Yeah, thanks for having me, glad to be here, this is a great to show and I am glad to finally be a part of this. And actually one of my commenters predicted this would happen and I actually this up in October, 2004 somebody posted a comment about I just discovered your blog you have been bookmarked, you should be on .NET Rocks!, I don't know if they've done a show on Human Factors, though, way back in 2004. So, I am going to have to email this guy and say, hey, your wish came true, I am finally on .NET Rocks! I have arrived.

Carl Franklin: Those were the Rory years I think, 2004, that was when we were sort of a little bit surreal and a little comedic and two hours long and so, it may have been why.

Richard Campbell: Actually it just took two years for us to get our act together enough to invite you. Maybe that's it.

Carl Franklin: Well, you know we had referenced your blog post on your site before, and I know I had done on HanselMinutes as well. So, what is Coding Horror? What's this all about?

Jeff Atwood: So, Coding Horror was my 2004 New Year's resolution and it started out like a lot of these things start with, sort of a half-hearted New Year's resolution to do something and in this case, that something was a reading list. So, the place I was working at the time, I didn't have a lot of other developers to talk to, that were, sort of loved software as much as I did. For a lot of the



people I worked with and not that there is anything wrong with this, software was just their 9 to 5 job, it wasn't really something that they took home with them, something that they thought it was really interesting to study but I did. So, I didn't really have any outlet, any avenue to express a lot of that. So, sort of the genesis of the blog and it's something I referenced in my bio is the recommended reading list. So, I had all these fantastic books about software development that I was reading that were just, to me it's just these watershed events of my career as an erstwhile professional software developer. And I just wanted really an avenue to talk about the book, to talk about software development, in a place where I could sort of have I guess, you might call it a reading club on the Internet, something I wasn't getting out of my current work experience.

Carl Franklin: And I noticed that *Code Complete* tops the list?

Jeff Atwood: Right, and I was really excited you guys had my idol, if you will, Steve McConnell on a few weeks ago, a month ago, that was a great show.

Carl Franklin: Well, I enjoyed that one immensely.

Jeff Atwood: Yeah and so, in my reading list, I won't belabor the whole list, but I think there is really three maybe four books, depending on how you look at it; that I find very helpful. I get to the point that I wish everyone I worked with on software had really read these books, I think they are really that important, '*Code Complete*' being the main one, of course, and then Krug's '*Don't Make Me Think*'.

(00:15:03)

Carl Franklin: I was just going to mention that as the second, it's a great book.

Jeff Atwood: Yeah, yeah, I also have '*The Mythical Man-Month*' but that's more of historical interest,, I think it's a great book, and I think it's the only truly classic book in software development. The grand daddy of any software book, is '*Mythical Man-Month*' but at this point it's more of historical interest, it's not going to really help you practically day to day.

Richard Campbell: And I think you said, you made the point here in your blog as well, which is the power of this book is that it's still relevant after all these years.

Jeff Atwood: Yeah, absolutely, absolutely and getting back to the I think to the human factors,

computers change so rapidly and that's what attracts a lot of people to the field is, you know it's a field where everything is changing all the time, so, if you don't like what's happening now, just wait five years, and almost everything will be different. But the one factor that really won't be different is the people, the way people work won't change very much in five years. So, that's sort of the one constant. And I think as I've sort of grown older and more experienced if you will as a software developer, I've found that more and more of the things that I find interesting are the people things because they are the only thing that I can invest time in, that will still pay dividends, five and ten years down the road. So, that's why Krug's '*Don't Make Me Think*' is an excellent -- it's based about people's behavior and usability and '*Code Complete*' although it's a book about how, to write software, it has huge, huge chapters, on personal character and just things you would think were out of scope for a software developer, but McConnell found them to be very relevant and it's just about the people behind the software had this huge profound impact on how long the software takes to write? How good the software is? Team dynamics, all those things that play a huge role in software are arguably more important than even the technology that you choose.

Carl Franklin: The comic strip on your blog, under '*Don't Make Me Think!*', perfectly illustrates that point.

Jeff Atwood: Right yeah, Web Design Funnies, that's a great one, because I've sat in meetings exactly like that, where it's sort of a last man standing filibuster on I believe we have to do it this way, no I think we should do it this way, and there is really no data, backing any of the instructions up, it's just, -- and you know the really sad thing is nothing really gets done because people aren't thinking about again software engineering. Let's make decisions based on data. If we can't decide which way to go then just lets do a little research and get some data points.

Carl Franklin: it's kind of like a an investment club I was in once, where nobody wanted to do any research, people would come in with doing the analysis of a particular stock and nobody would pay attention, people would fall asleep and then somebody would just pipe up, "I think, we should buy this", everybody "Yeah, yeah, yeah." No data at all. It's all feel, it's all like who's got the loudest voice in the room kind of thing.

Jeff Atwood: Yeah, it is depressing that it comes to that but I think software development in software engineering in general is a very immature field. You consider the birth of the



personal computer was basically in the early 70's maybe the late 60's. So, you are looking at a pretty short time span for the entire really micro computer as we know it and then software development is even less mature than that. So, I think part of this is just evolving everybody up sort of the evolutionary ladder of software and actually turning it into an engineering field. I mean, I don't think it'll ever be like the classic example, building a bridge, where you have know physics and these other bedrock assumptions that you can always count on, things are changing too rapidly but I do think there is a lot of opportunities to sort of go back and actually treat it like a legitimate engineering field.

Carl Franklin: What did you think about the Cooper's book, *'The Inmates Running The Asylum'*, that's on your list, I kind of like the whole idea of it, it was interesting, one of the -- his main manifesto in there is that users don't know what they want. And you know better than a user of your software what good software design is.

Jeff Atwood: Right, yeah Cooper's books are a little tough, I mean I love Alan Cooper, I think, you know he's sort of the grand daddy of Visual Basic, there's like that weird relationship there. He did the initial prototype of what became Visual Basic and presented it to Bill Gates, dascinating little bit of history. But I think, really with Cooper, he could be a little bombastic in terms of the point he is trying to make...

Carl Franklin: I sort of agree with you.

Jeff Atwood: Yeah, I would say, roughly half of a Cooper book is going to be like genius, and the other half is going to be kind of stuff like, I don't know if I agree with that, it's interesting to talk about it. I think you have to bear that in mind when you read Cooper's books. I do love them though and there is the reason that they are on the list because he's got a great very user-centered focus, that I think really helps a lot of developers.

(00:20:03)

Richard Campbell: I thought About Face was a better book, because it really focused on actually building a decent UI and the -- I've really never let go of his concepts of the different kinds of apps of the -- you know the sovereign app versus the sort of sidebar types and so forth. That to me -- when I read it years and years ago, really crystallized the thinking around different apps in different roles where people are -- sovereign app is the kind of app, that somebody is willing to learn the shortcuts on. I thought those were very fundamental thoughts about how we want our

applications to behave, and what you're allowed or should be doing as a developer, based on the role that outplays to the user.

Carl Franklin: So, the title of your blog Coding Horror suggests that you have some stories of coding horror, that maybe you shared on your blog, maybe you haven't.

Jeff Atwood: Well, Coding Horror is kind of a little bit of an inside joke. It's one of the -- that's a sidebar illustration from Code Complete, the original version of Code Complete, which I think is from like '93 or '94. I talk a little bit about the book, and the influence that book had on me as sort of a young developer. So, I graduated from college in '92, and sort of entered the field at that point, in terms of being paid to write software. So, I was "a professional", and this is really kind of free Internet. The Internet didn't really hit until maybe '95, '96.

So, as a young sort of inexperienced developer working in small businesses, where there is not really a lot of other developers to look to, to be mentors and so forth. I just sort of stumbled across this book, I was actually living in Denver at the time, there's this great independent book store called 'The Tattered Cover', giant independent book store. I actually found this book there, and it was just like revelatory, because it was this whole book about being a professional software developer, written in this very, very friendly human voice, not like this technical guru telling you the way things should be, but a rational clear voice. All this data, he had done all this research, and you know that's one of my pet peeves, is like, when we're talking about software, nobody usually has any data to backup the things that they're trying to talk about, and McConnell did, if you look at McConnell's book, there are all these data points, where he has done all this research about what he is talking about, and he's not dogmatic about it. He's not like, "Well, I did research and this is the answer, but here is the research I did, here is what I think, and here are my recommendations."

I just loved the book, and I think more than any other book, I call it the 'joy of cooking' for software developers. It's just a way to go back to, to him and say "Hey I love software, and I'm really interested in doing it the right way, as a professional." That to me is what Code Complete is about, and the revision I think is even better, it's a really significant revision that he came out with in 2004. So, if you have the older version, I do recommend upgrading, if you will to version 2.0 of the book.



Carl Franklin: I'd just to point out the Joy of Cooking is sort of like the Bible for the Julia Child generation in 50's and 60's.

Jeff Atwood: The word choice is very intentional there, where I do believe there's a certain aspect of enjoyment, you have to get out of writing software to really be good at it, you have to immerse yourself in it, and enjoy it. So, reading the book was just this big event in my professional life, and one of the most striking things about the book is that Coding Horror illustrates. Every time I saw that on the page, I would chuckle, because -- not because of other people's code, but because of my own code. To me, the minute you're amateur developer, you can tell, you realize that everything you write sucks.

Carl Franklin: Sucks, yeah. Oh my god I can't believe I wrote that.

Jeff Atwood: Yeah exactly. So, the minute you realize that, then you've crossed the threshold, now you're a professional developer, because you realize that you are your own worst enemy, almost all the time. Half of being a good competent software developer is just realizing that you're going to make just tons and tons of mistakes.

Richard Campbell: Learn to re-factor, this is what we're saying.

Jeff Atwood: So, it's like a lifestyle. Coding Horror is a lifestyle, it's your own code, it's not other people's code, it's really your own code. So, that's why I like it, it's sort of a clever in-joke about that.

Carl Franklin: You must have also had this experience where, you do code reviews on other people's code and knowing what you know, you find these issues just like you would find with your own, and sometimes it's tough for some developers to have that come to Jesus meeting about, "Oh, you're doing it wrong here, let's just fix it and move on."

Jeff Atwood: Right, and then the whole aspect of humility where, you are not your code. Being able to say, "Hey, I write a lot of code and a lot of my code really sucks." You've got to divorce yourself from the 'I am my code' mindset, which is I think very negative.

(00:25:02)

Carl Franklin: I agree.

Jeff Atwood: So, it's good to have some sense of humor about what we're doing, and our relative

competence levels. Nobody is really that good at software development. There's maybe like a handful of people in the world that are real geniuses at it, and the rest of us are just kind of struggling along, doing the best that we can, and I think that's another aspect of it as well.

Richard Campbell: I think you said it right Jeff. It's the humility, I was just thinking about another angle on being a professional software developer is being able to do a code review of somebody else's code, where your comment is something other than 'This sucks'. To actually see the value in the work, and to appreciate how that person was thinking when they created those things, it's the humility to recognize that, yeah there's more than one way to be successful at that, and my way is not always right.

Jeff Atwood: Absolutely, and I think this is where you get into talking to a compiler versus talking to another person, and I think for a lot of software developers, they got into the field because they are very, very good at technical things. Maybe even a little introverted, certainly that was true of me. So, in a way the computer is like an escape from people. Particularly in the early days of the Internet, although I think this is really, really changing as the Internet becomes very pervasive. The concept of a computer disconnected from other people, becomes very alien which is odd, because if you grew up with a computer in the early days, it was totally about isolation. Escaping into this world of the computer where one is one and zero is zero, and there's nothing in between, there's no grey areas.

So, I think a lot of what my recommended list is about is, is really the development of the non technical stuff. Being able to work with another developer and do a code review without like just ripping them to pieces accidentally, where you're acting like a compiler. You're saying this failed, this failed, this failed, your code is a failure. So, and another book, I'm really just going over the top five -- is Peopleware. So, Peopleware is really another watershed book in software development. It's cited all over the place, and I think really for good reason, because it really hones in on team dynamics, and how people interact, will have a bigger impact on your project, and your choice of technology or almost anything else from the project. I know McConnell has a book on software estimation, and he said that the three most important factors in estimating how long it's going to take to build whatever it is, that you're building or number one the size of it. So, if you're building this space shuttle, then it's going to kill you, but right under that, in terms of things that can impact your project and how long it will take, was team dynamics. Basically, personalities



and people, so that's the number two factor on any project, right after the size.

So, it's really important to have a handle on it, and I think for a lot of developers another one of the motivations for the blog was to get sort of better at communication with people and not the compiler, and sort of forcing myself to write blog entries, sort of help me crystallize my thoughts and sort of improve my communication skills. So, I believe that's tremendously important, and a lot of times my hope is, when people are reading a blog, they're motivated to basically start their own blog, and start doing some of these things and getting out there and amplifying what they're doing on the code side, with some complimentary skills, in terms of communication with their own team, with the outside world, with whoever, it's just that it's so essential. And for a lot of software developers, I certainly have met many that were technically super competent, really far more competent than I was, but I would have more of an impact on the team or the project, because I was just better at sort of expressing my position in a reasonable way, and just communicating with the team, and I always thought that was kind of sad and I really urge software developers to explore that.

Richard Campbell: The number of times the advice I have given to have really frustrated software developer in a fairly large organization, the only advice I gave was, "Write it down and email it, put it in a Word document, because it's 10 times more valuable in a Word document as an attachment, as it is in an email." Just the thoughts you've got, the concerns you have, that persistence of that document is incredibly powerful, and it really comes down to just kind of communicating your thoughts. The commitment you make to write that stuff down in a coherent form, so that somebody has some hope of understanding it, and then handing it around makes it almost unavoidable that you're going to get communication back.

Jeff Atwood: Absolutely, and I think too it's something you have to practice. Just like you would, as a practicing software developer, you gain skill incrementally over time as you do things. It's the same thing with communication; I mean if you're not writing anything down, you're not really going to improve your writing muscles.

(00:30:03)

You have to sort of make a commitment to do it to get better at it, and I think for so many software developers, they just fall into that -- I hate to say rut, but it really is kind of a rut of -- where they become deeply, deeply technical, and that's really not professionally what's going to help

them. What would help them more is to have some complimentary communication skills to go along with those good technical chop. Certainly with my blog I sort of made a commitment to -- originally I was to do one post almost everyday, but I realized pretty quickly that was a difficult call, and now my goal is to do one post for every weekday, not the weekend but five posts a week.

Certainly, when people are starting out with blogging or they're interested in this idea of improving your communication skills, that's one of the -- I think the most important pieces of advice is to make a commitment to actually some number of posts per week, whether it's one or even one a month, or something reasonable. But, just you got to practice it. You're not going to tell a developer, "Hey, write this down in a code here and email." What you're going to get back is probably not what you had in mind. So, it's something that I think developers need to cultivate and practice.

Carl Franklin: You know when comparisons of web development components come into play, vendors start tossing in clichés like complete toolset of controls, superior performance, empowering users and hosts of other buzz words, but at the end of the day, what matters most to you? The developer. For our friends at telerik, the answer boils down to simply getting your job done, like saving precious time by customizing stubborn controls that design time, or skinning new applications in no time, and how about no browser compatibility issues, that's a big one. Take the telerik Ajax offering for example, the product was designed to quickly get you up and running with this new, yet complex technology and it just works. Forget about writing tricky JavaScript, forget about making end to end modifications to your application. What's best is that you can count on a wide range of resources, sample apps, tutorials, active forums and of course telerik's renowned support team. After all, there is a reason why 89% of telerik's customers choose to renew their subscriptions. Experience that for yourself, by testing a trial version of the most reliable UI suite for ASP.NET at <http://www.telerik.com>

Jeff, you've been writing in your blog lately about things like Reddit and Twitter and patents. It's some interesting stuff there.

Jeff Atwood: Yeah, one of the things I find with my interest is that I really just love computers and the blog is kind of like my love letter, if you will. A very public love letter to the computer, to software and really the entire eco system around it. So, what I'm what you might call more of a generalist, where I have a lot of broad interests, like I'm really interested in the hardware, I like



building my own computers and so forth. That's not really for everyone, but it's just something that I find very interesting, and it's a part of the system, and I really enjoy exploring all those pieces. Certainly, I guess this has worked really well, because I have built up this rather surprisingly large audience, and I think being a generalist, I think appeals to more people, and this is probably a repetition of the theme earlier, where when you come very narrowly technical, it becomes difficult to sort of expand your base and really talk to people, that may not be as technical as you.

So, certainly a lot of topics that I really like to explore. One of my greatest tips you might say, is this article, "Separating Programming Sheep from Non-Programming Goats." So, a lot of times when I write blog entries, I don't really know what the reaction is going to be. I've written blog entries that I thought were going to be really interesting to people and they were essentially ignored. Then, on the other hand I've written just little pieces where I felt like I got to post something to the blog, I would just write something up in two hours, and it becomes this sort of huge Internet -- let me be careful how I say this, a huge Internet phenomenon in the sense that a very popular post on my blog. Maybe not world famous if you will, but just a lot of hits, a lot of comments, a lot of track-backs and things like that. This post was one of those things, where I was like, "Oh, this is kind of mildly interesting." I just wrote it up, and the premise of this post is really just recapping a study that was done in academics, where people were teaching software development. This group of researchers sort of expressed this frustration, and actually the quote here is that between 30% and 60% of every university computer science department's intake fail the first programming course.

(00:35:11)

So, you're talking about students that are saying, "Hey, maybe I want to learn software development," so they express some level of interest in software development, and more than almost half of them on average sort of wash out and really don't grok programming for some reason. It has really frustrated the people teaching, because they see this pattern repeated like pretty much the entire time they're teaching software development. I thought that this was really just fascinating, because this parallels sort of what I do in my professional life, where you have some developers that just get it, they live it, they understand it, for whatever reason they just jive with software development, and there are some that just really don't get it. I have certainly seen this professionally, and I was sort of intrigued that before you even become a

professional software developer, there's this huge divide of people who get it, and people who don't get it.

Richard Campbell: It's kind of staggering to me, the idea that the first time you would program is when you saw a course in college to go program. I mean all three of us are obviously the other species that program -- computers and programs were such a drive for us that we went and sought out that experience. It wasn't part of the educational process that made us want to program; it was wanting to understand the machine.

Carl Franklin: Scott Hanselman and I talked about this and we referred to your post about this, which was the FizzBuzz test, using FizzBuzz to find developers, and it was, "Why can't programmers program?" That was your blog post.

Jeff Atwood: Right, so that's getting into people who would ostensibly apply for a software development job. So, you would think these people have already crossed the chasm, right, they should theoretically understand how to program.

Carl Franklin: By the way, I've shrinksterized the 'Separating Programming Sheep from Non-Programming Goats' at shrinkster.com/o5e if anyone wants to read that. But, yeah it is amazing when -- basically the premise of this post is that people who would be applying for a software job were asked to do the FizzBuzz test, where they just need to write a program that spits out a series of numbers and then for -- I can't remember exactly what the premise is, but certain numbers get printed out as fizz or buzz depending on whatever it is. It is a simple mod and division kind of stuff, and a whole series -- most programmers can't do it basically.

Jeff Atwood: Right. Well, what I found shocking was that, this is a very rudimentary test. I mean, this is like -- say you're hiring truck drivers. So, where is the gas pedal, where is the brake? Can you shift gears? I mean, not can you complete an obstacle course in your truck, but just do you know how to operate a truck? I just had a very appalling -- and it was really echoed by a number of different people sort of in the Blogosphere if you will, that, where they had seen this exact phenomenon. People coming in that really literally could not complete a FizzBuzz, and I thought going a little bit deeper that the programming sheep and non-programming goat, sort of gives you sort of a sneak preview of why that happens, and the students -- it was a very simple test they came up with to try to figure out if they could predict who was going to succeed and

who was going to fail earlier. It's a really simple test, there's an example in the blog post, but it's basically all about assignment. It's about constructing a mental model, and it doesn't even have to be a correct mental model. But a coherent mental model of how assignment works, just by looking at the command.

The funny observation here, that I found very rang extremely true for me, was that you didn't have to be right, but you had to be able to form this very rigid authoritarian view of how assignment works, and you had to be totally rigid, just like a computer is. Computers, they don't care what we're thinking, they're always going to mindlessly plod down whatever path it is they're going on, and the ability to put yourself into that mindset was so crucial to sort of being able to be a programmer, was to accept the inflexibility of the computer, and I found that really fascinating.

Carl Franklin: The thing I have seen before when I put people on the spot is the jeopardy effect, where you just sort of freeze up, and you can't see the forest for the trees, because you're trying to over think the problem. You're thinking that somebody's throwing such a curveball at you that you can't possibly figure it out, so your nerves get all -- you know you get all tense and nervous and stressful and that just has a devastating effect on your ability to think. I'm not cutting anybody any slack, because come on FizzBuzz, easy.

(00:40:00)

Richard Campbell: Yeah, it should be reflex, and the funny part is, I found developers who interviewed well, and couldn't FizzBuzz, and I found more developers that interviewed horribly and FizzBuzzed fine. Or I've had a guy who was just appalling in his interview, his resume is barely literate, and it's remarkable he got as far as me, but I give him a coding problem and he rips it up, like he lights. Suddenly he is on the whiteboard and he is all over it, and I'm like, "Okay, I could deal with the social issues, to get the talent."

Jeff Atwood: I think that's great, because that's exactly the divide that I hope that Coding Horror is about, it's about bridging that divide. People who are really good at coding, but just can't make it through and interpersonal skills required to an interview are just beyond them, but then people who -- and this is also a problem. People who are good at talking about software development, but don't necessarily have the technical chops to actually do it. So, you want to bridge that gap and have a balance of skills, I think as a professional software developer.

Richard Campbell: So, the one thing that I've really taken to heart, both as a consultant helping other companies hire them, and for my own company's doing hiring, is I always ask with the resume, "Send me your favorite piece of code," and it taught me a bunch of things about a person right away. If they couldn't come up with a piece of code, they're probably not a programmer. Every programmer I've ever met within our series of programming has some code they love, but it was also the best possible ice breaker, for even the most socially uncoordinated developer. That piece of code they really like, when you get them to talk about it, you can see it in their eyes. Off they go, doing their thing, talking about their chunk of code, plus I had a chance to review the code ahead of time and say, reading that piece of code let me know a lot about them, "Well, if you wrote this code, then you must know the following things." My FizzBuzz test for them was then, script it around their code sample.

Jeff Atwood: I think that's a great strategy, and that makes a lot of sense, because that gets back to the Joy of Cooking. I mean, if you don't enjoy the code, if you can't find it -- I always found it weird, we would do a similar thing, we'd say, "Okay, send us a code sample to start with," and you sort of get all these back pedaling of, "Oh well, all our code is proprietary, I can't really send you anything," and that always just seemed like such a giant copout to me, because when you write code, I said earlier, you are not your code, so there's this process of separating your ego from your code. Whatever you write, that's like your baby. If you can't find code that you care enough about, to just take it out and share it with -- some fragment of it with someone.

Richard Campbell: Yeah, make it non-proprietary.

Jeff Atwood: That's such a deep problem, that I would almost not want to interview somebody at that point, if they couldn't find some code that they put together, they were actually excited about. And another thing that feeds into -- is one thing I like about the blogging strategy, is the blog sort of becomes your professional business card if not resume. Occasionally, I do post code on my blog, so certainly anyone is interested in what I do and the type of code I write, should be able to find that pretty easily on my blog, so having a blog means you sort of have a professional calling card already out there. So, it's interesting, when I started to work at Vertigo, one of the things I liked -- one of the many things I like about Vertigo is that Scott Stanfield, our CEO actually read my blog prior to the interview. This is the first company that had done that. Where I sent out my blog, it's like, "Okay, here is my resume, here is what I do, and here's my blog." Scott had



taken the time to actually read through it and really liked it, to the point that we had conversations about some of the topics that I had brought up on my blog.

Carl Franklin: Jeff, can you pick a recent blog post that you felt was really good, and can we get into the meat of it?

Jeff Atwood: Sure, so I have a few that I like and some of the more popular stuff that -- and as I said it's hard to tell what's going to be popular, so you just kind of throw stuff out there and then some hits and some doesn't. One that really seemed to resonate was 'The Programmer's Bill of Rights.' I don't know if you guys have seen this.

Carl Franklin: I have seen it.

Jeff Atwood: So, what motivated this post was that at Vertigo, sort of I have two identities, I have sort of a Clark Kent Superman thing going on, where with my blog I have a sort of secret identity, although it's not really secret anymore, a lot of people know about my blog, but for a long time it felt that way, and then in my day job I actually do go out and interact with other developers that companies outside Vertigo, largely around team system, Microsoft team system. It always pains me, when I would go out and I would see developers working in what I considered terrible, terrible conditions where they didn't even have the basic equipment to do their job correctly. Depending on where you stand on capital investments for software developers, you might not agree with this but I think as much as we pay software developers, it's crazy to send them out there without 10 grand worth of equipment that they're going to need to be truly effective, and yet I saw it like all the time.

(00:43:13)

Because everywhere I went, it was like crappy mice, crappy keyboards, monitors that were 17 inch CRT's from like 1998, and things like that.

So, The Bill of Rights is actually a list of environmental things that are really easy to fix, because a lot of developmental problems are really hard to fix. Team dynamics, or the product you're working on doesn't make sense, but these are things that you can throw money at, and fix. That's what I got so frustrated about, and just stuff like having two monitors in this day and age, the productivity benefits of two monitors is I think indisputable and very inexpensive.

Richard Campbell: There is a very fundamental message around equipment in general, which is this, "We value you" message. We're giving you

the best possible gear, because we think your success is important, you need to demonstrate that value.

Carl Franklin: There is a mentality however, especially in the -- from people who have worked their way up that, you got to pay your dues. If you can generate good code on crappy equipment, then you earn the privilege to have good equipment. I mean, that's sort of an old school mentality, I think is what's happening there. Don't you think?

Jeff Atwood: Yeah, I think so. I mean I think paying your dues is different than having the proper equipment to do your job, I think that's so fundamental; it really is truly a Bill of Rights. I mean it's something where, if you're a developer and you don't have these things, I really think you should go in and demand it from whoever calls the shots. I think it's well within your rights to do that.

Carl Franklin: Two monitors is actually going to make you a better programmer, it's not a luxury, it's not for playing Quake.

Jeff Atwood: Not at all. In fact, they've done all these studies to show -- to me it's overlapping Windows. As much as we talk about GUI, the problem is overlapping Windows are like this huge productivity tags. I mean manipulating Windows is pure exercise; it's just not useful work.

Richard Campbell: Straight up lost time.

Jeff Atwood: Yeah, straight up. I mean exactly, I would easily believe that 10% figure that I've seen in the study is just based on 10% less time I spend dragging Windows around, manipulating them, trying to make something visible, because you used to have more space. You don't have to have overlapping Windows. So, again I view that as just very, very fundamental. Just having a reasonably fast PC, I think compilation time is waste of time. I mean, every second I have to spend staring at the compiling dialogue. It's time that I could be writing code, and the really thing is that it kills you. It's like death by a 1000, 10 second delays. You add up those 10 second delays over time and that's maybe 30 minutes of time. So, it seems really small, "Oh, what you're complaining about, it would be 5 seconds versus 10 seconds," I think it's a forest and trees problem there where they're not seeing the cost of a computer is just nothing relative to the labor cost. It's craziness.

Carl Franklin: How about the situation in which a software developer who is making \$60-100,000 a year is paid to write a programming tool that



clearly exists in the market for \$500. I see this over and over again, and in my town -- I mean I don't have to -- I can step out on my front porch and see somebody that's doing this, and they burn thousands and thousands and thousands of dollars on this guy to write a sub par programming tool that they could just as easily purchase.

Jeff Atwood: Yeah, that's very frustrating. I know, I have participated in projects like that where you just do the math, you take the number of people working on it, multiply by an average salary, and you just divide that by the number of users you have, and it's just a staggering, staggering number where there's just no way that can actually make business sense.

Carl Franklin: Yeah, I've talked to the developers in this case, and they're like, "Well, I didn't have anything else to do." What? What? How about taking a book and reading it, how about learning something? You don't have anything else to do? Come on.

Richard Campbell: You know, one of your ending points in the whole Bill of Rights thing, was the quiet working conditions. My solution when I was the only developer in the office, was I hung a sign on my door that was my -- called the 'Programmer's Productivity Matrix' and it was measured in interruptions per hour, and optimal performance I discovered was 0.25 interruptions per hour. So, one interruption every four hours was my maximum to be a productive programmer.

(00:49:56)

Jeff Atwood: Right, and I think it's great to highlight this stuff, and my hope is that people, -- particularly people who are starting around in their careers as professional developers, can read that and realize, that I thought the phone and the interruption were annoying but I have a fundamental right as the software engineer to ask for that stuff, to be, if not taken away entirely just toned way down. it's part of learning the craft of software development. This is learning what tools do I need to actually do this stuff effectively and some basic stuff about expectations on working condition. Now, on Carl's point of higher level, does what you are working on actually making any sense? That's a really deep problem, and I know, I think, it's that if you think about the number of projects, you've actually worked on, that actually resulted on shippable, useful software that made some impact on the world, I think again it's depressing. To think about how little of the code, you write actually results in something meaningful, to even a really small subset of users. it's really challenging, I think,

that's why as a craft, as a person who practices the craft of software development, you have enjoy the act of really creating software, I think that has to be enough reward in and of itself...I mean....

Carl Franklin: And that's the goal, right? I am reminded of Billy Hollis, who says, most programmers think their job is to write code. it's not, it's to provide solutions, that's your job.

Jeff Atwood: Absolutely, absolutely, and I think, that's a big picture thing that takes a while for developers to sort of, to sink in and then to sort of marinate in it for a while and actually work on some failed project. And I have another blogpost, that I talk about, the Importance of Failure in Software Development, because you have to realize that failure is sort of the default mode of software, better or worse statistically, most software projects fail, but I think that's okay. I think, what you have to think about there is analyzing, like why did they fail and again the divide between amateur developer and professional developers, realizing hey, my code sucks. Similarly on projects, you know realizing that my project failed, but here is why it failed, like self awareness is really what I am all about and that at the code level as well as at the project level, like I felt like, you know the retrospect is, I participated in for the project, even if I had to sort of force them on the powers that be were very powerful because you got into, why are failing? What are we doing wrong? Rather than just going out and repeating those same failures modes, it's what I call the Gilligan's Island problem, and this is totally taken from Steve McConnell by the way, by the way I can't claim credit for this as every week there is some crazy new scheme to get off the island, whether it's building some piece of software that already exists, or just something that doesn't really make sense, ultimately, and yet they go through and hilarity ensues and at the end of the episode, they are stuck on the island for another week. So, how can we really get off that island and I think, it's about self awareness, it's about reading books, like if your job is to be a project manager, what books have you read, that sort of explain what a project manager actually does. And going back to the recommended reading list, so, having some contact...

Carl Franklin: Are referring to Success through Failure, which is your post from May, 24th 2005?

Jeff Atwood: Possibly, I have several on failure, sadly, but I think...

Carl Franklin: Well, Jeff I just want to put out the Shrinksters for these that one is at shrinkster.com/o5i, Oscar five indigo, and the



programmer's bill of rights is at Oscar five golf, shrikster.com/o5g.

Jeff Atwood: So, I think failures are kind of like a badge of honor. I think, if someone can come in and tell me, I worked on a bunch of projects, one was successful and then I had a bunch of it that were failures and here is why I think they failed. I think that is much more helpful, than somebody who comes in and says, everything I have worked on has been a spectacular success.

Carl Franklin: Yeah, I agree.

Jeff Atwood: Number one is just, it doesn't seem -- it doesn't gibe with my understanding of the industry and two, you know, failure isn't necessarily an evil in itself, it's what you do with that that really makes it work.

Carl Franklin: Preach on, brother.

Jeff Atwood: There is a tenacity that comes out of -- I mean you have to give up eventually but you just have to be incredibly tenacious because, computers ultimately are super frustrating. I mean, everyday, my computer defeats me in some small, but important way, something I am trying to do.

Richard Campbell: And it makes you agree with it by popping up a dialogue that made you say, okay.

Jeff Atwood: Yeah, exactly so, it's about the sort of the reality of failure and we live in a highly, highly imperfect eco-system. As I mentioned earlier, software development is a very young field still, I know it's crazy to think of that, I think a lot of developers, particularly as they try to become older developers and get some years of experience under their belt, get the idea that, well this is how it's going to be and I think the way you have to think of this is we are still in -- if you think of like say, the history of automobile, just to pick sort of a random comparison.

(00:54:04)

we are still in probably like the 1940s where the whole story has not even remotely been told, on the history of software, so it's exciting because I think, we have a part and I think still the early history of software development and that's why, with Coding Horror, it's exciting to me to be a part of that and think, that people are actually reading what I am writing and thinking about some of these issues and how to evolve as a software developer.

Carl Franklin: Jeff, do you have kids?

Jeff Atwood: Not yet, not yet we're kind of working on that.

Carl Franklin: So, one of the things that I find, the more enlightened parents who have the smarter kids have, just among people I know, is to let them fail. To set up a safe environment to let them try things and fail and to let them know that, that's how you learn and a kid can learn more from a broken dish than the parents constantly trying to say, watch out, watch out for that, watch out for this, watch out for that.

Jeff Atwood: Absolutely.

Carl Franklin: It's just something that I have picked up on it and there is a parallel here because ultimately raising kids is sort of the ultimate software project.

Jeff Atwood: Right, so going back to your original question was, you have seen development teams working on just ridiculous projects, that can still have value. So, developers in that, can realize what cycle they are in and next time hopefully recognize that and mitigate it, then I think ultimately it's a step forward, even if that product is ultimately kind of a waste of time for everybody involved, as long as they've learnt from it involved as sort of software engineers. I think that's the lesson you have to take out of it. And you have to enjoy what you are doing, so that there is always that bedrock to go back to.

Carl Franklin: Jeff, tell us about the post, has Joel Splosky "jumped the shark?" We have brought up jumping the shark a couple of times because of this.

Jeff Atwood: Yeah, sure no problem, first allow me to put a disclaimer out there, I think Joel Splosky is a brilliant writer, and I think, almost everything he has to say, is worth reading. What really motivated this post was there is this perceive disconnect between the things Joel was saying that software developers should do and here is to how we run Fog Creek Software which is Joel's company, and then sort of some of his later posts, he started peeling back the curtain a little bit on some of the stuff they were doing and it really didn't seem to gibe with a lot of the advice he had given other developers and I just was a little frustrated by that, to where I felt like Joel had been giving this great advice, and then basically not following his own advice, and it's just frustrating.

Richard Campbell: Well, the whole Wasabi thing was incredible and your line was perfect, you couldn't possible have heard it but that was the sound of 50,000 thousand programmers heads simultaneously exploding.

Jeff Atwood: Right, we invented our own language, I mean this is a canonical example of what not to do? I mean, if a developer ever came to you, you guys, I said hey you know what I think we should do, I think we should write our own language.

Richard Campbell: Right.

Jeff Atwood: Right, when is that ever the right answer. I mean, I think, Joel had some reasonable reasons for doing it at the time but I think they quickly became legacy baggage and it's just, it's frustrating and I'll tell you what, it's exactly why it's frustrating, because if someone as smart and talented as Joel can fall into that trap, it just shows you how, easy it is to make the decisions...

Richard Campbell: How are us, regular mortals going to survive.

Jeff Atwood: Yeah, exactly and you know, Joel is a smart guy, he'll obviously ride this out and I am sure it's not a problem for his company, but even in Joel's post, you can sort of see some hesitancy when he talks about it. He is kind of not -- he is not basically acting like Joel at that time, he is kind of throwing it out there and kind of hesitating and that's not Joel. Joel is very much like, here what I am doing, here is why it's awesome.

Carl Franklin: Let me just clarify a couple of things, first of all the post in question is shrinkster.com/o5j and jumped the shark, tell us what that is a reference to?

Jeff Atwood: So, jumped the shark sort of has become a meme. I think, it's a reference to a Happy Days, episode. I didn't actually see his episode. I was never a big Happy Days fan, but I guess it was later in the life of the show, and I guess at one point the Fonz, you know Fonz, actually jumps, somehow gets on a motorcycle, wearing shorts which -- the Fonz, wearing shorts is already weird and to make it even weirder, it actually jumps over the tank that contains the shark on his motorcycle. So, it's become short hand for "has become irrelevant". Like I have evolved into...

Carl Franklin: Yeah, that was the beginning of the end for Happy Days, that's when the show began to tank.

(01:00:00)

Jeff Atwood: Yeah, yeah, you are doing something so crazy just to get attention that you have forgotten about your core competency as a

show and I think that's somewhat true of Joel. Although again, I have tremendous respect for Joel and I always will and I think, Joel's book for example is excellent and in fact, I am about to add his UI book to the recommended reading list. So, I am a reluctant critic of Joel and I wish, he'd write more too. I think, over time, he has kind of gotten away from writing a lot of content and I think it's just too bad, because Joel is really a brilliant writer and one of the things that's interesting about Joel, is Joel has, and I don't have the citation for this but Joel is talking about the value of functional specs and it was interesting because when Joel was talking about okay, you just got to write functional specs that people want to read. That's very easy to say, when you are Joel Spolsky, because Joel Spolsky is probably one of the most talented software writers, writing. He is super entertaining. I mean, I can give his writing to my wife and she doesn't care about the technical stuff but she would be entertained by Joel's writing and she would read it all.

Carl Franklin: And he has written some great stuff. I mean, just we got to keep saying that, he's written some great stuff.

Jeff Atwood: Yes, absolutely, absolutely and but sort of the myopia of Joel is that, when he says, oh you just got to write functional specs the way I would write them. This is incredibly difficult. I mean if the bar -- if you have to be as good of a writer as Joel Spolsky, to make functional specs that are actually people want to read, then you are never going to get there and I think that's again just Joel not being able to see outside Joel, unfortunately. It's great advice but it's very particular to Joel.

Carl Franklin: Well, Jeff we are coming down to the end of the show, is there anything you want to plug, or say hi to, or shout out about, or talk about a toy or something you've seen on the web recently?

Jeff Atwood: No, I don't think so, I just want to thank everybody that reads my blog. It's become this really great community and I want to thank the people that comment there and a lot of the comments that people leave, really do feed into my thinking on the blog. It's not as if I am putting a message out there and then I have these comments, that I kind of ignore but those messages affect my thinking and the whole point of doing this is to have that two-way dialogue so it actually understands a lot of the stuff going on in software development. I certainly do not have all the answers and I really appreciate everyone who participate on the blog.



Carl Franklin: And we really appreciate having you on the show. It was good talking to you.

Jeff Atwood: I enjoyed it.

Carl Franklin: All Right, and we will see you next time on .NET Rocks!

(Music)

Lawrence Ryan: .NET Rocks! is recorded and produced by Pwop! Productions - providing professional audio, audio mastering, video, post production and podcasting services, online at <http://www.pwop.com>. .NET Rocks! is a production of FranklinsNet - 'Training Developers to Work Smarter' and offering custom on-site classes for Microsoft Development Technology with expert developers, online at <http://www.franklins.net>. For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.