



.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 # 126

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



Rocky Lhotka discusses CSLA.NET

August 12, 2005

Our Sponsors

CoDe

component developer magazine

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



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



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! 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! Crack open Rocky's latest word processed thought-transfer service and listen up, its 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 # 126 with guest Rocky Lhotka recorded live, Friday, August 12th, 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 telerik r.a.d. controls – "The most comprehensive suite of components" for ASP.NET Development online at www.telerik.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 whose most difficult decision last Thursday was boxes or briefs, Carl Franklin.

Carl Franklin: Thank you very much. Thank you and welcome to another stellar episode of .NET Rocks! I am Carl Franklin and I'm here on the East Coast of the United States and as always my partner on the West Coast of North America, Richard Campbell. How are you?

Richard Campbell: I'm having a great time, man. Crazy, crazy week.

Carl Franklin: What's crazy about it?

Richard Campbell: Well, it started on the weekend actually. I popped down to Kim Tripp's place for a party.

Carl Franklin: Yeah.

Richard Campbell: And, in the middle of the party, Don Box walks in, plunks himself down and holds court. The discussion of the day, which took over the whole party and was with...

Carl Franklin: And why wouldn't it.

Richard Campbell: Yeah, and why wouldn't it. With, the discussion of the day was to whether or

not relational databases were actually necessary anymore.

Carl Franklin: (Laughing) Come on.

Richard Campbell: I mean, Kim Tripp's place, right. I mean, we're talking about the sequel queen, and there's Don's topic of the day. He wants to talk about where if we have a transactional file system and we have index server, what do we need SQL Server for anymore.

Carl Franklin: Wow.

Richard Campbell: Yeah. Subtle, huh?

Carl Franklin: Yeah.

Richard Campbell: It was actually a great discussion. A tough discussion went on for hours. But it was really made you think and I guess that's what Don was after.

Carl Franklin: You just can't get Don and Kim in the same room with alcohol and not expect something like that to happen.

Richard Campbell: (Laughing)

Carl Franklin: I actually wish I could have been there. That sounds like a lot of fun.

Richard Campbell: It was a lot of fun and Bob Beauchemin was there and Ted Neward, too. I mean a lot of great folks together.

Carl Franklin: Wow. When did they finally kick you out?

Richard Campbell: Ted had his sons over as well and they were playing Xbox. So we ended up towards the end of the party playing Burnout 3 crashing fire trucks into large intersections not the most intellectual pursuit but just a howl of fun. So we were out of there after midnight and Ted was nice enough to put me up for the night and I drove back in the morning.

Carl Franklin: Well the reason Jeff said that joke about last Thursday because it was my birthday.

Richard Campbell: It was, too. Happy birthday, buddy!

Carl Franklin: Yeah. I'm 24, man.

Richard Campbell: (Laughing) Or something.

Carl Franklin: (Laughing) Yeah, 38. Good Lord.



Richard Campbell: There you go you and me both.

Carl Franklin: Everybody's asking me, what did you do for your birthday? I'm like I'm trying to lay low man because after 21 birthdays become a reminder of the fact that you're getting old.

Richard Campbell: Yeah. You really go down to only decade recognition.

Carl Franklin: Yeah, exactly. So, I'm really not into the whole celebrating the birthday thing and, quite honestly, that's why I forgot to get my wife a present last year and the year before and the year before. So you...

Richard Campbell: Right.

Carl Franklin: It's not that I'm absent-minded or anything. It's just that I'm doing it as a favor to her so she doesn't have to be reminded of how old she's getting.

Richard Campbell: Did she actually buy that?

Carl Franklin: What do you mean, buy what?

Richard Campbell: Actually buy that that's why you didn't get her a gift?

Carl Franklin: That is why I didn't get her a gift. I'm doing it as a courtesy to her, man.

Richard Campbell: (Laughing)

Carl Franklin: I expect the same. (Laughing)
Good Lord.

Richard Campbell: Well, we're having a big bash for your 40th.

Carl Franklin: That I could do. That I could definitely do.

Richard Campbell: Of course, I'm going to have to last up until, like, 150 shows to do that.

Carl Franklin: Yeah. Well I've been getting a lot of e-mail lately that we haven't been too technical lately because we've been doing a lot of high-level things about new features coming out, new technologies that are coming out.

Richard Campbell: Yeah. We're headed towards a launch here. So that's sort of the focus, right?

Carl Franklin: So here's a good example of this. Andy Andersen sent us this e-mail. Hi, just want to say that you have a great show and I'm trying to catch on listening to most of them but some

comments. The current show was great but I would like to see a new separate show just in the way you with Monday. What I would like to see is a show that is even more technically focused. The current .NET Rocks! Has lots of talking about non-coding stuff like conferences, personal events, gossips, and that's fine and interesting. So what I would see is a hardcore .NET Rocks!, where you for an entire episode, focus on one or two products or technologies and real in depth, instead of general overviews like what you have at .NET Rocks!. Like having a dedicated show just to focus on Rocky's CSLA framework. (Laughing) Hearing from users of it, from Rocky's strengths and weakness really focused, in depth focus. You could also focus on issues like code generators, frameworks, solutions to serious issues, data access caching, large sites, site structures, best practices, remoting. Andy. Well, what do you think of that, Richard?

Richard Campbell: I think we're pretty technical already, but you're right. We have been doing more generalized shows lately mostly because we are headed into another conference season and that's what's on everybody's minds right now.

Carl Franklin: True. I'd like to think we do a pretty good mix and maybe we, like this guy is pointing out, Andy, is saying, you're right, lately it has been pretty much non-technical. And it's very interesting that he said that we should make a show just to focus on Rocky's CSLA framework. That, in fact, is what we're doing today.

Richard Campbell: There you go.

Carl Franklin: Yeah. So and I also want to tell you that we lined up Rocky to be the guest about this before I had even read this e-mail. So it was fortuitous for Andy that this week show is going to be right up his alley.

Richard Campbell: I would throw one other comment along those lines, which is, it's actually really tough to do detailed technical content in this format. It's got to be a little broad.

Carl Franklin: Yeah, it does.

Richard Campbell: I'm not going to read code to you over the air.

Carl Franklin: Right. And there is room for a good mix. But anyway, I would just suggest going back listening to all the shows and I'll think you'll see that we have a good mix. So anyway let's go ahead and introduce our guest, shall we? Now I want to tell you that our guest was going to be somebody different. Steve Schwartz. But something came up and he couldn't make it and



we're rescheduling him for November, I believe. So I called up Rocky and I said, "Hey Rocky, what are you doing tonight?" Would you like to do the show and we wanted to get a little bit more into your CSLA framework and a bit more technical. And he said, "Sure, that sounds good." So let me just introduce Rocky Lhotka for probably about the fifth time, I think. Rockford Lhotka is the author of numerous books, including the Expert Visual Basic.NET and C# Business Objects books. He is a Microsoft software legend, Regional Director, MVP and a member of the INETA Speakers Bureau. Rocky speaks at many conferences and user groups around the world and is a columnist for MSDN online as well. Rocky is the principal technology evangelist for Magenic Technologies, one of the nation's premier Microsoft Gold Certified partners dedicated to solving today's most challenging business problems using 100% Microsoft tools and technology. Will you please welcome again, Mr. Rocky Lhotka? How are you, sir?

Rocky Lhotka: I'm doing great, Carl. Thanks. I wouldn't miss your birthday show for the world.

Carl Franklin: It's not my birthday show, man. When I go to Chili's, I tell people it's not my birthday, all right. I don't want the clapping and the cake and stuff, man. Let's just forget I even said it.

Richard Campbell: (Laughing)

Rocky Lhotka: I still think we should have probably sung Happy Birthday.

Richard Campbell: I think you might be right, Rocky.

Carl Franklin: No. No, please. Please no.

Rocky Lhotka: (Laughing)

Richard Campbell: I mean you know what, there's no point in doing it because he'll edit it out anyway.

Carl Franklin: That's right. We have the power of the edit.

Richard Campbell: There you go.

Carl Franklin: So what have you been up to, Mr. Lhotka.

Rocky Lhotka: Oh, I have been doing all sorts of fun things. I've been playing with CSLA.NET and I've been playing with .NET 2.0 and Visual Studio 2005. And in fact I've even been off playing with datasets and seeing what the non-object world is up to these days.

Carl Franklin: Yeah, the non-object, object world. So have you come to any new revelations in that regard?

Rocky Lhotka: Well, a lot of a new dataset features and capabilities are interesting and if they work the kinks out of them before release, they could be useful to a lot of people. But I got to say that all of the things that I'm doing over there with datasets are just reinforcing to me the benefits of objects. (Laughing)

Carl Franklin: Right. And to anybody who isn't familiar with what Rocky is talking about, he has produced a series of books going back to Visual Basic 6.0 or maybe even before that. Was it before Visual Basic 6.0?

Rocky Lhotka: Oh, all the books. The first one was Visual Basic 5.0.

Carl Franklin: Visual Basic 5.0, okay.

Richard Campbell: Wow.

Carl Franklin: So Rocky's area of expertise is writing object frameworks with business objects and using them for data structures and for modeling the real world and binding in all aspects of your applications. And I guess he's really slayed a lot of the dragons that people run up against and at one point he decided, there's a lot of stuff that we're reusing, I am going to just make a framework. And, so, he built this framework of base classes that you can use in Visual Basic.NET now called CSLA.NET and that framework comes with his book. So it's not a separate product that he sells. You get it for free if you buy the book. And so, contrary to that is the dataset centric model of architecting data that exist in the middle tier in the business objects, where you either create objects that have datasets as members or you try to derive from dataset and stick your own logic in there and so I guess that's the issue. It's the business object model versus the dataset model.

Rocky Lhotka: Yeah, really I look at it as that the world really comes in two flavors. It's either you're striving to be object-oriented in terms of managing your business logic or you're pursuing more of a data-centric or really trying to extend the relational model in the memory in your computer. And if Don Box is right, maybe the relational model is dead. I don't know.

Richard Campbell: (Laughing) But you know, a lot of people build apps that way. They start with a table and work outward.

Rocky Lhotka: Well, yes they do. And in fact there's no doubt that the majority of software is



created and written using a data-centric model either using data tables and datasets or there are other similar toolkits out there that -- alternate generators that create datasets, or things that are like datasets. And people will find that to be very intuitive and I think the reason is that the relational mindset is ingrained in our thought process. It is hard to find a programmer that can't at least bring data into the third normal form intuitively.

Carl Franklin: Yeah.

Rocky Lhotka: And it's kind of natural to extend that, I think, into your actual software development, too.

Carl Franklin: I had a woman come up to me at Dev Connections. I did an entire N-tier application architecture session in one day, which was all based on datasets. And she came up to me afterwards and she said, "I talked to Rocky about how we should architect our app". And he offered his book and his solution told us about it and we did it, and it just took a really long time. It was a lot of extra effort. Now we have an application. And so she was there wondering whether or not -- she's sort of peering over the fence looking at what the other side is doing and that's really true. I mean that's the payoff or the trade-off, right? In the dataset world so many things are just automatic but you give up some control. And with the business object world, you spend a lot more time on architecture and being more explicit in what you're going to do but it takes longer and there are more chances for you to do something stupid. Isn't that true?

Rocky Lhotka: Yeah, I think there's probably some truth to that because there's always no doubt that you go down an object-oriented road, you do end up writing more code, at least initially compared to the data-centric approach.

Carl Franklin: Yeah.

Rocky Lhotka: But I'm convinced, in the long run, that the object approach is off in making your code more maintainable, more agile.

Carl Franklin: Yeah.

Rocky Lhotka: And actually, over the life of your system reduces the amount of code and effort involved.

Carl Franklin: Right. And it's always a decision point for people, which way they should go for any given application. Some applications are prone to some architecture and others are prone to another, I would imagine. Don't you think that's true?

Rocky Lhotka: I think there's a lot of truth to that. And people often ask me what kind of applications CSLA is suited for and what kinds it's not. And you know, my primary background comes from creating interactive business system, things like order entry and inventory management systems things where, there is a user sitting there interacting with the system.

Carl Franklin: Yeah.

Rocky Lhotka: It's commonly called OLTP.

Carl Franklin: Right.

Rocky Lhotka: Online Transaction Processing and so a lot of my energies, in designing CSLA, went into to supporting that particular model.

Carl Franklin: Yeah.

Rocky Lhotka: And, at the same time if you look at a typical business system there's a lot of, what I guess you'd probably call maintenance screens, where you're just basically editing trivial little look-up tables and there's virtually no logic there.

Carl Franklin: Right.

Rocky Lhotka: And at that point, it's really hard to say that the dataset is worse than object because there's nothing there anyway, right?

Carl Franklin: Yeah, there's nothing, that's right.

Rocky Lhotka: But it's when you get to the actual, the interesting parts of the application, where you're managing your inventory or whatever, where there's complicated relationships and logics then objects really shine.

Carl Franklin: Yeah.

Richard Campbell: It's really more of a mature development model. It's this recognition that software is going to persist over time and that different people are going to work on it over time. That needs are going to change over time. I mean, you've got to be able to support all of that.

Rocky Lhotka: Well, I think really that's why I recommend that if you go down the object approach that even your maintenance screens get created with objects. Because even if it is a little extra effort, that way you've got a consistent model and it makes it easier to do a long-term maintenance.

Carl Franklin: So. Yeah.

Rocky Lhotka: You don't want to bring a new developer on six months or two years from now



and say, "Hey, we need to make this change and, by the way, you're going to have to learn like two or three different programming models to work on different parts of this thing."

Carl Franklin: Yeah. Well, the question that's on everyone's mind is that, well I already have the .NET framework. I already have tools for creating properties and methods and classes. What all is your framework going to give me and that's really what I want to dive into here for the next, oh I don't know, forty minutes or so, is, what are the base classes in there and what are the problems that they overcome?

Rocky Lhotka: Yeah, I mean that too is a pretty common question because .NET has, depending on who you talk to, like eight or ten thousand classes in .NET alone and it's pretty valid question to wonder why you need more. (Laughing)

Carl Franklin: (Laughing)

Rocky Lhotka: But I do think that there are some things that -- well basically if you look at the .NET framework, it's a generic programming platform as it allows you, theoretically, to do almost anything. And when we sit down to create a business system our focus is narrowed and we blatantly ignore the vast majority of the .NET framework.

Carl Franklin: Yeah.

Rocky Lhotka: And the reality is that there are parts of it that are really useful like reflection and remoting and some of the security components, data binding that you want to use but you really don't want to have to write that code over and over again to do those things and that's where, the framework comes in.

Carl Franklin: So, let's dig into it. What are some of the classes?

Rocky Lhotka: Well, I think the best way to approach it is to kind of look at it from a feature perspective.

Carl Franklin: Sure, goal-oriented. I'm all over that. I'm all about goal orientation, man. (Laughing)

Rocky Lhotka: All right. Well, first I guess it allows you to create objects that both understand how to interact with the database, in other words, how to manage persistence, get their data in and out of whatever kind of database you might have, relational or otherwise. And, at the same time, your object ultimately has to be used by a Windows form or a web form.

Carl Franklin: Right.

Rocky Lhotka: And so, if you think about it that way, these objects almost have like a split personality. And let's tackle the UI side first. And when you look at that it is my view that you want to minimize the amount of code in the user interface and in Windows or web. And the reason is that the user interface is by far the most volatile part of a typical application.

Carl Franklin: Yeah. And binding, especially in .NET, as we've seen just stretching out beyond the grid, it gets pretty complex.

Rocky Lhotka: It gets, it certainly can and well the interesting thing is -- we were talking about whether objects were more work or not and prior to .NET, they were a lot more work.

Richard Campbell: Definitely.

Rocky Lhotka: You couldn't do data binding objects at all.

Carl Franklin: Right.

Rocky Lhotka: And now in .NET 1.1 and 1.0 Windows Forms data-binds the objects pretty nicely. Web Forms, it's okay. And when we look forward into .NET 2.0 and some of the features that are coming, the Visual Studio 2005, all of a sudden data-binding objects becomes really, really sexy. I mean, way beyond anything that we've got today. So, it's pretty compelling. So part of what the framework does is, it adds extra support for data binding.

Carl Franklin: Right.

Rocky Lhotka: And .NET by itself can data-bind essentially any object. You combine controls on Windows or Web Forms into an object with no extra work. But what you'll find is that you won't get the behaviors you really want. In other words, sometimes when you're updating the object, it won't show up on the screen...

Carl Franklin: Exactly.

Rocky Lhotka: In order to overcome that, you have to implement a set of interfaces. On your collections, there's...

Richard Campbell: What are those interfaces? Let's just get into it.

Rocky Lhotka: Well, I hope your readers get into the details here. On the collections there's an I-binding list which is an interface used by data-bindings to both allow the UI to control the way the list works to some degree, in other words,



filtering and sorting. But then it also declares an event that allows the collection to tell data binding when anything has changed inside of its collection. And that interaction is very important and an I-binding list is what does that.

Carl Franklin: And that is in 1.0 and 1.1 we're talking. But this has nothing to do with the binding list in 2.0, right?

Rocky Lhotka: Well, no it actually...

Carl Franklin: I know it does, but you're talking about 1.0 and 1.1 here.

Rocky Lhotka: I am. And in 2.0 that interface continues to be important. But in order to use it, I think most people are going to use the new generic of binding list of T.

Carl Franklin: Yes.

Rocky Lhotka: But that's all, all that is, is a generic wrapper around the same interface.

Carl Franklin: Very cool.

Rocky Lhotka: And, which is important for people to realize because if you've been writing objects and collections for the last two or three years, they'll continue to work in .NET 2.0 because the interface is the same. It's just now there's an easier way for you to write your own collection. Now, that's the key one for collections.

Carl Franklin: And doing that turns out to be difficult in 1.1, right? Implementing that interface, especially in VB, is that right?

Rocky Lhotka: Yeah, so (Laughing) there is that. I suppose I should make it clear that my CSLA framework exist in both Visual Basic and C# versions. But, and the only real difference between the two is that the Visual Basic version has a tiny bit of C# in it. And the reason that it has that is because when you go to declare an event in .NET 1.0 and when you start using the object say for Windows Forms, if you ever try to serialize the object, which has been converted into a byte stream, it'll actually, the serializer will attempt to serialize the Form as well. It'll follow your event, link, the delegate link, it'll follow it right up to the Form. And to overcome this in .NET 1.0, only C# has the syntax to pull that off.

Carl Franklin: Yeah.

Rocky Lhotka: And in .NET 2.0 both Visual Basic and C# has the same syntax. And I've got a couple articles on my blog actually showing the code in both VB and C# on how to do this and this issue goes away. But, yes, there's a little

complexity in terms of declaring a list change event. It's actually declared in a C# base class and then the rest of CSLA inherits from that or at least the collection portion does and I use the standard .NET pattern, where the list class raises the event but then exposes a protected method called On-list changed.

Richard Campbell: Yeah, I do that religiously in my components. This might be a nice little tangent to go off on here about this pattern. It's kind of confusing when you're looking at just the form for example and you look at all those overloads and you see all these virtual subs that say on and then the name of the event and because they look like they relate directly to events and well, it's very natural for a programmer to wonder what's the difference between handling that and the overrides and actually handling the event.

Rocky Lhotka: I think actually there's probably two reasons why this is the way it is. One of course is purely technical. When you declare an event in a class, it's not possible for a subclass to raise that event. The event can only be raised within the class that was declared.

Carl Franklin: Yeah.

Rocky Lhotka: And yet many times you want to allow events to be raised in derived classes. So there's the pattern that Microsoft came up with was to do exactly this where you have on such-and-such method and when you call that, then that triggers the event.

Carl Franklin: Right. So it's called internally by the base class to raise the event and the sub does no more than raise the event. So this is the way that you can modify the behavior of controls in Forms is by hooking an event and if you want to eat the event and not allow it to go any further, you can just do nothing or do whatever you want to do in place of it. And then you can all my base dot on whatever it is on enter, on leave, on changed or whatever and pass your E parameter and then you can do some code after. So you get control before the event. You call the event yourself by calling the base class implementation and then some code after. Right?

Rocky Lhotka: Absolutely. And then the other thing -- the other reason that this occurred is kind of a broader issue in that .NET was designed to be an object-oriented framework. And yet it has to exist in an event driven world, meaning Windows. And when -- especially when you look at the Visual Basic heritage, where things are very much centered around the event model they needed to carry that event model forward. Well, partially because it is a natural fit for Windows but



partially because the majority of Windows Developers are Visual Basic Developers and so it is very important that they make that familiar model carry forward.

Carl Franklin: And at the same time, I would warn Visual Basic programmers, who are just getting into the Visual Basic.NET not to jump to handling an event as the first way to solve a problem because that often leads to lots of duplicated code. More often than not you want to be handling these virtual subs inside the object itself that you want to change the behavior of.

Rocky Lhotka: Well, certainly there are faster, too. Overriding the method is typically faster than handling the event. But all of that said, it's important to also look forward to Visual Studio 2005 and the whole partial class concept and realize that the use of events is much easier to use events of partial classes than it is to try and override methods. And so, I think while your advice, is good for today, it might not be so good necessarily in Visual Studio 2005.

Richard Campbell: Not that I disagree with you, Rocky, but why?

Rocky Lhotka: Well, the primary issue is that in Visual Studio 2005 what happens or like a Windows Form or a Web Form, is that the IDE is generating a whole bunch of code for you. But it's much of the same code that it generates today, but they're generating obviously some new and different things, too. And all of that code is generated into a file that you don't normally see. So it'd be called like Form1.designer.vb and that's a hidden file by default. And your code, for the Form, would go into Form1.vb. And you won't any longer see the hidden code region, right?

Carl Franklin: Unless you want to.

Rocky Lhotka: Unless you want to. But normally it's just not even there. And the thing is that you're coding, your code that you're writing gets merged with the designer file before it gets compiled. Literally, they're just going to append into each other.

Richard Campbell: Right.

Rocky Lhotka: And, so if Visual Studio, for instance inserted a method or override a method in the designer code, you can't have a method of the same name in your form code, too because it doesn't know how to merge that. You'd actually get a duplicate method, compiler error. And so, in order to avoid that issue, the normal approach is that your, your Form1.vb will end up primarily just handling events.

Carl Franklin: Right. Yeah, and then, just to summarize if any one of the files in the partial class is already overriding a virtual sub, you can't re-override it in another piece of that class. And breaking out, a call into another sub, I suppose, is okay. You know what I mean? You could always just go in and either modify the existing partial class, the override, or call up, break it out into a sub in your new class. But the problem is a lot of the reason you have these split classes is for code generation because the designer generates the code. And if that code ever needs to be regenerated you've lost those hooks and you don't even know because you don't see it.

Richard Campbell: But whereas if you have multiple event handlers for the same thing, it doesn't matter at all. It's going to work the same.

Carl Franklin: Right. And an event handler is going to use a delegate, which is a little bit of added overhead as opposed to just overriding a virtual sub. I don't think this is in the grand scheme of things going to matter much unless you're calling those events in a high degree of repetitiveness.

Rocky Lhotka: No, that's right. Well, and in here, I think you need to draw a line specifically between Windows and web development because if you're building a Windows system all of these events are firing specifically on the user's desktop. And so they're only competing with themselves. Whereas, on the web the overhead of the event firing, arguably can be a bigger deal because it's a shared server resource.

(Music)

Carl Franklin: Well if you listen to the show you have heard me talk about ASP.NET tools from telerik at telerik.com. They have recently released a new version of their r.a.d. control suite Q3 2005. And I would like to tell you about it. Telerik r.a.d. control suite is the most innovative and comprehensive toolset for ASP.NET development allowing professionals to build web solutions with the UI richness and responsiveness of desktop applications. The latest milestone release Q3 2005 is the first on the market to bring full XHTML 1.1 and accessibility compliance with WCAG "Level A" and Section 508, thus enabling developers to build standards-compliant web applications easier and faster than ever. Added to this are key updates to four of Telerik's most popular products – r.a.d.editor, r.a.d.grid, r.a.d.treeview, and r.a.d.rotator. R.a.d. control is also available with an annual subscription options for all updates and new components added to the suite within a year of your purchase. Hey did you know that



.NET Rocks! Website was done with the telerik menu. That's right if you are using menu on the left hand side, you are using telerik's products. So, go check them out at www.telerec.com.

So let's get back to CSLA that is one of the first things that I show, is the binding base. I can't remember what you call it, bindable business object base or something like that?

Rocky Lhotka: Yeah.

Carl Franklin: I do show people how to use that to make their own custom collections instead of using the standard collection base. And it literally is one line of code changing the inherit statement in a bindable collection. So that's -- and as you mentioned before, you actually had to write that in C# because of the way to split out the delegate and serialize it properly because the Windows Forms isn't serializable. And then the next part of your framework that would give you the most, I don't know, the coolest feature, would be what?

Rocky Lhotka: Well that's collections. The other main avenue of course is that you might create a single object like in order or customer. It's just not a collection. It's just a single entity. And so the framework has a class called business base that you inherit from. And business base -- I've spent a quite a bit of functionality both data bindings in this case interfaces like iEditable objects and IData error info and it raises yet another event that's declared in C# indicates that the object data has changed.

Carl Franklin: You mentioned a couple interfaces, let's just clarify where those interfaces -- are they yours or they .NET?

Rocky Lhotka: .NET. iEditable object and IData error info are both, I believe out of the component models, system.component model.

Carl Franklin: And so what's the benefit of using your base class over just implementing that interface and plugging in then?

Rocky Lhotka: It really comes down to good object oriented programming. If you are creating a class that most of that contain business code for a customer. Good object oriented programming would say that your class should only have code dealing with customer behavior. And so if you end up also implementing iEditable object and IData error info and some of the other things that CSLA does it's quite realistic that you could have more plumbing code just to support .NET stuff then you would code dealing with being a customer. And that's just bad object oriented programming.

Carl Franklin: It seems to be though, if these things are interfaces iEditable object for example, then the reason that their interfaces because the implementation is going to differ from object to object, is that not the case?

Rocky Lhotka: Well you might think so. But in this case we are able to make all of the business objects fundamentally work the same. And in fact with iEditable object the reason is this, that interface exists to support in place editing in a grid. And it allows the grid to tell your object that it's about to edit you and that the user either press the escape, the row should reset itself or the user arrow it up or down in which case the row should keep any changed values.

Carl Franklin: So that seems to me to have nothing to do with the object itself.

Rocky Lhotka: Well you wouldn't think so. Except that it's my view that objects should know how to undo themselves and in particular this is because your typical Windows application has a cancel button on the form. And so the user can come in, change a whole bunch of information in your object and then click cancel and what happens at that point?

Carl Franklin: Yeah.

Rocky Lhotka: So you might say well maybe we discard the object.

Carl Franklin: Right that's not a good idea because there might be other references to it.

Rocky Lhotka: Exactly. And so what you really want to throw the object from how we store it with all this previous values. And so that's another key feature of CSLA because that all of the object is inherent from business base and all of the collections that inherit from business collection base all understand how to reset themselves back to a previous state if you want them to. But that offers what I use for iEditable object.

Carl Franklin: Okay, you did mention this in an earlier show that we did that you did this with reflection and as opposed to say serialization and de-serialization, which would create a new object, which would break your existing references. So you did this with reflection you actually enumerate all of the properties and you save the original versions of those properties much like the dataset does actually.

Rocky Lhotka: Yes that's true. Although I actually support N levels of undo, which for a lot of UIs doesn't matter but several times they have ended up creating UIs that have nested model forums or other really sophisticated models and



so being able to actually edit the object at different stages and then undo it to a certain point, whereas the dataset only has one level of undo.

Carl Franklin: Rocky I have a specific question about a particular issue that anybody who has done any binding with text boxes knows about, which is when you've got text boxes bound and you have got some way to move the cursor through a set or a collection like say a list box or something and you just make a change in a text box and then you want to go ahead and commit that change but save the changes, let's say by pressing a button lets say you're just doing a maintenance screen. The edit won't have taken place, isn't that true, because just by moving off the text box you don't end the current edit. So hasn't been committed unless you actually move to another record?

Rocky Lhotka: With the dataset that's true but with object it's not.

Carl Franklin: That's interesting.

Rocky Lhotka: So where is this the data binding works is that at least with CSLA object and as soon as you tab off from a field the value goes from that field from the control into the object immediately.

Carl Franklin: So what are you hooking a changed event like every time you press a keystroke are you hooking last focus?

Rocky Lhotka: It's only when the control loses focus, this occurs.

Carl Franklin: Okay. Is that something that you are responsible for or is that just a part of the binding behavior?

Rocky Lhotka: No it's actually just part of binding behavior.

Carl Franklin: Interesting.

Richard Campbell: But you can see in the data model why they don't do that, they want to wait for all the changes to happen in the entire record before they go writing it back.

Rocky Lhotka: In the dataset that's the case, yes?

Carl Franklin: Yeah.

Richard Campbell: Yeah.

Carl Franklin: And it's interesting when it's in a grid you get it when you move from cell to cell it

end the current edit. But if your text box is bound you have to do an EndCurrentEdit on the BindingContext. And I have gone into situations where I have created a text box that hooked the on leave event and then checked to see if there was a binding to the text property and if it was ended the current edit and I am into all kinds of problems with binding with that text box because there are times when I wanted that behavior and times when I didn't want that behavior.

Rocky Lhotka: Now I got to say this that luckily none of these issues happen with objects.

Carl Franklin: That's really very cool (Laughing). That's great, no that's great.

Rocky Lhotka: Yeah, so with an object as soon as you hear the controls lose the focus the value gets put into the object and then if you click your 'apply button' or whatever then the code behind the 'apply button' is typically maybe with three lines of code that just tells the object that you like the changes and that it should save them. And the object is responsible for saving it's of into the database and a way you go. And the other final main feature is the thing called broken rules, which is changed quite a bit. Actually when I wrote the books essentially you can think of the framework in the books is being version 1.0 and the framework is currently at version 1.5.

Carl Franklin: Okay.

Rocky Lhotka: So there have been a number of enhancements since the book came out. And the current broken rules actually allows you to do essentially inform at all of your rules as a set of message they conform to a specific delegate signature. And rules in this case really what we're talking about validations behaviors like a field is required or max length where one field has to be bigger than another. And one of the consistent things with validation rules is that they all ultimately evaluate to true or false. And so it's totally realistic to create a generic delegate signature and then not generic like in the 2005 stand. But it is a delegate signature for your message that can implement literally any rule and so that's what I have done in the current framework and so you can implement all of your rules as a set of message that return true or false. And then you register those rules with the object as basically when the object gets substantiated you loop through and then say well the name field is required and has a max length of fifty and the description field is required and has a max length of eight thousand and your start date must be greater than the end date and you register all of these rules and then from that point on to a large degree the object is track off whether its in a valid state or not on your behalf.



Carl Franklin: And it's like what they are doing in datasets with the state the record state or the row state whether it's been modified and added, unchanged, deleted.

Rocky Lhotka: Right, to some degree it's like that only this is more sophisticated because it's that not only implementing those basic concepts but also implementing your custom business logic.

Carl Franklin: Yeah.

Rocky Lhotka: And so as the user tabs off a field, if the field and say blank a field and tab off that empty string is immediately put into the property. The property by being set figures the appropriate validation rules not all of them either just the ones dealing with that property. And then you are assuming that's a required field the CSLA framework actually sets your object into a message is valid property default and through the IData error info, interface is able to notify the UI that this particular property is in error so an error provider, a little red mission mark appears.

Carl Franklin: Right, that's cool. Do you have a similar concept to the row state or an object state? Like what if you delete an object from a collection, do you actually delete it or do you just set a marker that says, "This one should be deleted when I go to update my database or do you just leave all of that stuff up to the user, up to the programmer rather?"

Rocky Lhotka: No, the framework directly handles that and it's actually none of the above, business collection base it ultimately inherits from System.Collections.CollectionBase, which is a collection but it also internally contains its own private collection of deleted items.

Carl Franklin: Oh cool.

Rocky Lhotka: So when you delete an item from a Collection what happens is that item is physically moved out of the real collection into this private deleted collection. And that way when you ask an object to undo itself it's able to restore any deleted items back into the original collection. Similarly it keeps track of objects that are added to the collections that if you reset or undo the collection that can restore it and it can remove those newly added ones.

Richard Campbell: When do the deleted once gets cleaned up?

Rocky Lhotka: The deleted once gets cleaned up when you save the object. Because obviously they actually need to hang around and so you

ask the object to save itself and when it is in the process of saving itself to the database, that way the objects know to delete themselves or delete their underlying data and at that point, that the objects physically go away in memory as well.

Carl Franklin: That's cool. So Rocky, you've got a new version coming up for Visual Studio 2005 languages .NET 2.0, right?

Rocky Lhotka: I am working on it as we speak.

Carl Franklin: So what are some of the things that we can look forward to in that version?

Rocky Lhotka: Well, I am obviously going to take advantage of generics. And so we look at the language enhancement for both Visual Basic and C# coming up is the additional generics and reusing like I mentioned earlier, binding lists of T radically reduces the amount of code that you need to write or create a collection. And the CSLA will have a new base class called business list base, which we use binding list of T and so you inherit from that now instead of business collection base. And it looks to me like it should cut the amount of code you write down by about 70%.

Carl Franklin: Wow!

Rocky Lhotka: Yeah, the greater collection.

Carl Franklin: And that's the code that you write or that they write.

Rocky Lhotka: That they write.

Carl Franklin: Okay, but isn't a lot of code savings just because of the fact that it's a generic, not necessarily the fact that it's a CSLA.

Rocky Lhotka: Oh, absolutely. I am making no bones about the fact that I am exploiting the new features that's the whole point.

Carl Franklin: Right. And the thing is though if you are looking at CSLA in 1.1, that this is one of the best reasons to move along with it. Is that you get all these great features and all the features of 2005.

Rocky Lhotka: There are other things in 2005 that we'll tap into, for instance the new data binding capabilities are just extraordinary. You be able to literally drag your CSLA business class on to a form and have it either draw a grid or a detailed form for you.

Carl Franklin: Yeah, that's fabulous.



Rocky Lhotka: That stuff works beautifully and there's some minor tweaks the CSLA to make that work a very, very trivial. And so when you really look at it from a CSLA perspective, I was able to convert the CSLA 1.5 code just by itself to run in .NET 2.0 in an afternoon. So a lot of what I have been doing is trying to figure out ways to preserve as much backward compatibility as possible, also offering up the ability to tap into generics to reduce the amount of code for a collection or for a business class and it happened to the new data bindings.

Carl Franklin: The price of software success though isn't it?

Rocky Lhotka: Yeah, we haven't really talked about this but one of the things that CSLA, one of the new features that it's got a concept in it called the DataPortal that allows your object to persist themselves to the database. And the DataPortal abstracts the network. So the one in the book uses remoting and obviously that's wonderful today but looking forward we want to be able to use Indigo and even today some people want to use web services or Juval would tell you that if you are not using enterprise services then you creating toy application.

Carl Franklin: Well, that's Juval

(Laugh)

Rocky Lhotka: I just had to get that dig in there.

Richard Campbell: You just had to.

Rocky Lhotka: And so I have actually been working a lot on the DataPortals. In the CSLA 2, the DataPortal is going to be configurable so you can tell it do the data access locally or to do it remotely through remoting, remotely through web services, remotely through enterprise services or remotely through Indigo. And the really cool thing of course is that like moving to Indigo, where people using CSLA, there won't be any code changes required because the DataPortal will entirely abstract it.

Richard Campbell: Rocky, our friend John Bristow asked a question in the chat room too. He wondered if you found a performance improvement using generics in CSLA 2.0.

Rocky Lhotka: Well, I can't say that I have done any extensive performance testing. So I can't give a definitive answer to that. However, other people who have done testing using generic collections have seen some pretty dramatic performance improvements. And the new business list base because that uses a generic under the covers does in fact hold everything in a

strongly typed list. So I would expect that CSLA is going to see the same -- some of the figures I have seen are like 60 times performance improvement. in dealing with collections and lists. But yes I'd expect to see a lot.

Carl Franklin: Here is another question for you. Can you use the class designer in Visual Studio 2005 using your CSLA.NET base classes? Does that work well together?

Rocky Lhotka: Unfortunately no, not really. And the reason is that the class designer is really designed around simple classes and CSLAs classes that you create tend to be a little more complex. I know maybe I am speaking a little too harshly. The reality is that you can in fact use the class designers but I haven't found personally that it seems to save me a whole lot of effort. I am a bigger fan of code generators.

Carl Franklin: And how about productivity enhancers like CodeRush.

Rocky Lhotka: Things like CodeRush or CodeSmart or other similar tools, allow you to create customized templates are tremendously valuable. As an alternative to code, some people just don't like code generators. I understand that and, but being able to create a full blown CSLA business object template and inserted in like taping four characters or something like some of those tools like CodeRush 2.0. It's really, really cool.

Carl Franklin: Would you be into getting together with the CodeRush people to put your templates in there?

Rocky Lhotka: I think that'd be very interesting.

Carl Franklin: That would be very interesting, wouldn't it?

Richard Campbell: That would be cool.

Carl Franklin: Maybe we could hook you guys up. What code generators have you used and do you like? And you don't have to consider this a plug or anything but I mean seriously what ones have you looked at and what ones do you like?

Rocky Lhotka: Well, I have looked at and a lot of people use CodeSmith. And as far as I know CodeSmith actually comes with templates with CSLA 1.5. And what is the other one, LLBGEN. I haven't looked at but lot of people speak highly of.

Carl Franklin: Declare it as another one.



Rocky Lhotka: And all right what else, Magenic the company I worked for has one that we use as part of our consulting service offering. But my primary criticism with all of these code generators it tends to be the same with most object relational mapping tools, is that they tend to focus on, that you start with the database schema and create the object based on the database. And great object oriented design should be behavioral and so it should be really based first on the database. Obviously data is important but it should be based first on the behavior following from your used case or your agile XP stories with the user whatever techniques you are using to get the work flow in business process from the users. It's that behavior that should be first. And so I guess disappointment was all of the current generators that I am aware of is that none of them do that. They all focus on the database first.

Carl Franklin: Interesting. What else do you have planned for? Any new features of, I know that there is a lot of rewriting of code to support .NET 2.0. But what about new features that .NET 2.0 makes possible besides the obvious generics, maybe sub features that because of generics are possible.

Rocky Lhotka: I am not necessarily planning on radical enhancements to the framework.

Carl Franklin: Okay.

Rocky Lhotka: And part of it is I actually blogged about this a little bit. Part of the reason is that I have used CSLA primarily as being part of my book rather than in a product. And so basically the book is already pretty big, like around 800 pages. And the publisher is telling me that I don't want to make it bigger.

Carl Franklin: Stop writing, goddamn it.

Rocky Lhotka: So anything I do has to fit in my existing page counting. So I am rearranging the book structure a little bit. You are cutting out some of, like the current book in chapter three there is an overview of reflection and remoting and some things and I am just cutting that out. I will assume people by this point have a clue about how those things work and that allows me to focus more. But really a lot of the "enhancements" that somebody might see are because the book will be about CSLA 1.5 plus generics, plus the new DataPortal, plus the new data binding capabilities.

Carl Franklin: The DataPortal sounds awesome. I mean the fact that you can use Indigo to pass objects around in no matter what the transport that sounds really cool.

Rocky Lhotka: All the flexibilities there, it's tremendous.

Richard Campbell: I am just thinking from a testing point of view is going to be killer. It's going to be a little try out all these different transports with your same app and see how it behaves.

Rocky Lhotka: Yeah, exactly. The flexibility there is – it's pretty exciting I think. And so on one hand you might say that it's disappointing maybe that there's not going to be radical changes, but on the other hand even though I try not to look at CSLA as a product. The reality is that a lot of people use it as is or with minor modifications. And if I radically change it for a 2.0 version, then what is that do to all those people that are using the thing today. And so I feel that I should have at least some level of consistency. And I think that's the hallmark of a good framework anyway that it should survive at least some reasonable period of time. My Visual Basic 6.0 framework, which is quite different from the .NET 1.0 but really started in Visual Basic 4.0, believe it or not that is on the Visual Basic 4.0 project when I created the framework and I wrote the first book in VB 5.0 and so that the original VB framework just survived for about eight years and was fundamentally the same during that entire time. And personally I hope that CSLA.NET is similar in that, oh yes, it should be enhanced and have some nice things added over time but one would hope the core of it remains consistent as long as the .NET platform itself doesn't change out from under us.

Richard Campbell: That's really seems to be like a proof of proper abstraction.

Rocky Lhotka: Yes, I agree.

Carl Franklin: Sure, and architecture too. Yeah, the fact that you have got so many implementations out there using this thing and all happy campers just in of itself means -- I am curious is to why this isn't something that Microsoft would want to design into their own component name space.

Rocky Lhotka: I don't know the answer to that. I can speculate Microsoft for start Great Plains and then subsequently started working on the Microsoft business's framework or foundation or whatever that's good for. And which is a pretty lofty goal what they are trying to accomplish with that project. And CSLA or similar frameworks or tools like this are really I think more tactical.

Carl Franklin: Or more fundamental too, don't you think? I mean you are making something that can be used in a broad variety of applications.



Rocky Lhotka: Yeah, if you look out there, there is not a universal agreement on, what kinds of architectures are good by any means. All you got to do is prove the BlogSphere for ten minutes...

Carl Franklin: Just listen to our last ten shows.

Rocky Lhotka: There is no agreement and so for Microsoft you adopt a specific interior architecture would be a pretty big step on their part and they would risk even if .NET supported all of the other architectures, the fact that they pick any one as being their favorite would have the potential of reducing the number of people that want to use .NET.

Carl Franklin: Well there is the question of do, by supporting it to the inherently send the message that this is their favorite or do they just add another tool to the tool chest, which is the way I would see this.

Rocky Lhotka: A lot of people would see at the other way I am afraid.

Richard Campbell: Even people inside of Microsoft too, I could disuse being a big battle internally to get something like that written.

Rocky Lhotka: Well, and the other thing I think politically right now, object orientation and especially the concept -- really distributed computing is moving toward today with whole service oriented idea of all you passed around is raw data in message packet. And the idea of actually moving objects around the network will meet violent opposition in certain quarters. And it's just not trendy. I think it will be again everything goes in the cycles and eventually we hard ugly limitations of passing raw data around are going to become clear. And people are going to come back and go, oh maybe we should look at this object thing again.

Carl Franklin: You can bet Rocky, this is in the end of the discussion where you talked about this at the DevConnection show that you did with Bill Vaughn and you are not saying that SOA is a bad idea, you are just saying that there is a lot of hype about it because it's so new. And a lot of people don't even understand what it is and what the benefits are. And they can't agree on what it is. But you are going to find that I think we are going to be coming back to trying to nail down what kinds of systems lend themselves to a service oriented architecture and what kind of systems do not? I think that's the real value in a discussion that we can give our listeners. Over time hopefully we will all discover the answers.

Rocky Lhotka: I absolutely agree with you.

Carl Franklin: Yeah, well man, we are coming to the end of the show and I want to ask you what the coolest thing is that you have downloaded lately? What's the coolest thing man?

Rocky Lhotka: The coolest thing I have downloaded lately well, I got to say that at the moment I didn't exactly download this. I am hooked on Battlefield 2.

Carl Franklin: Really? (Laughing)

Rocky Lhotka: I have been pretty much every night spending at least an hour or two mostly getting blown up.

Carl Franklin: For some reason I just can't picture you sitting in front of a computer while the world crumbles around you're going die, die, die.

Rocky Lhotka: You do, I am really good in a tank. I got to say that.

Richard Campbell: I say you are a Sherman guy.

Rocky Lhotka: Yes.

Richard Campbell: I was always Panzer II partial what can I tell you.

Rocky Lhotka: Battlefield 2 is all modern type stuff. So it's a little more if you are current in some of those.

Carl Franklin: And you have been listening to adventurous at the Anti-Monkey Brigade right on Mondays!

Rocky Lhotka: Oh I have and I love that. That is great fun.

Carl Franklin: Yeah Mark and I are specifically out of all the group. Mark and I really been doing a lot more the sort of audio theatre, radio theatre kind of stuff and really takes a lot of time to put together so that's probably not why we only have ten minutes spurts here and there but boy, that's a hoot. All right well thanks Rocky for coming on the show. I got to thank you on behalf of myself and the listeners out there, Geoff in the sound room and Richard Campbell. Thanks again. You are always welcome here.

Rocky Lhotka: Well, thank you. I appreciate the opportunity to talk to you guys and obviously I love this topic. So I had a great time. Thank you.

Carl Franklin: All right, I can't wait to see CSLA.NET 2.0. In the mean time you can check out Rocky's latest stuff at hotka.net. Thanks, Rocky.



Rocky Lhotka: Thank you.

Carl Franklin: All right will, see you.

Geoff Maciolek: .NET Rocks! can be found online at www.dotnetrocks.com and at msdn.microsoft.com/dotnetrocks. .NET Rocks! is edited each week by Geoff Maciolek that's me and Carl Franklin who is also Executive Producer. All music heard on .NET Rocks! including Toy Boy, the theme song is created and produced by Carl Franklin and Franklin Brother's band. Carl never sleeps. .NET Rocks! Is produced for franklin.net by PWOP Productions, "Providing professional audio and Podcasting services, online at www.pwop.com. PWOP – it's time to get your impact back.

(Music)