HTTP://www.dotnetrocks.com

Carl Franklin and Richard Campbell interview experts to bring you insights into .NET technology and the state of software development. More than just a dry interview show, we have fun! Original Music! Prizes! Check out what you've been missing!

Text Transcript of Show #640
(Transcription services provided by PWOP Productions)

Gael Fraiteur is Still PostSharp!
February 24, 2011

Our Sponsor

HTTP://www.telerik.com/

Lawrence Ryan: .NET Rocks! episode #640, with guest Gael Fraiteur, recorded live Saturday, February 19, 2011.

[Music]

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

Carl Franklin: Thank you very much and welcome to .NET Rocks! I'm Carl Franklin here in New London, Connecticut, Richard Campbell in Vancouver, British Columbia. Hey, Mr. Campbell.

Richard Campbell: How are you, my friend?

Carl Franklin: I'm well. Today's show is a little interesting. It's going to sound a little bit different because it was recorded on Saturday at the New York City Code Camp where I was and we talked to Gael Fraiteur about PostSharp.

Richard Campbell: Right, right, and I was still back in Vancouver. I actually did the engineering and captured both your lines. It's just funny that you and Gael were together but have to sit in separate rooms to do the recording.

Carl Franklin: Yeah, it's funny. Anyway, it's a good show and we'll get to that in a minute but first let's talk about Silverlight and Better Know a Framework.

[Music]

Richard Campbell: All right, my friend. What have you got?

Carl Franklin: All right, I want to talk about binding expressions, okay.

Richard Campbell: Okay.

Carl Franklin: You know, if you have the value of a property you can set the value explicitly or you can use curly braces and use the binding keyword.

Richard Campbell: Right.

Carl Franklin: If you use the binding keyword then you have a whole bunch of properties that you can set on this binding like the element name is the name of the XAML element that you're going to bind to.

Richard Campbell: Uh-hmm.

Carl Franklin: And the path is essentially the property of that element that you want. There are a couple of other really important ones but a cool one is the IsAsync Property.

Richard Campbell: Oh. What does that do?

Carl Franklin: IsAsync, lets your set a value that indicates whether the binding should get inset values asynchronously.

Richard Campbell: Oh, very nice.

Carl Franklin: So if you're binding to something that is going to call out to a web service or to another service that might take a long time, you can say IsAsync and your UI essentially won't freeze while it's going to freeze that value. Now you might say, "All right, but what's the default? You know, I want to show something while there's also a fallback value property that you can set."

Richard Campbell: Nice.

Carl Franklin: Yeah. So you set the fallback value property, then you set IsAsync and you bind it and you can go off and do something asynchronously in the element that you bound to.

Richard Campbell: Bob is your uncle so to speak.

Carl Franklin: And Bob's your uncle. And there's a really good example in the documents, just look up Binding.IsAsync property.

Richard Campbell: Nice.

Carl Franklin: You'll find that on MSDN, there's a really good example.

Richard Campbell: That's good.

Carl Franklin: So Richard, who's talking to us?

Richard Campbell: Here's an interesting email for you and I'm bringing it up because I want to talk about this actually. "Hi, Carl and Richard. I love the show. You guys do great work. This is not a question directly related to .NET, but I'm really interested in your thoughts on why Microsoft's CEO Steve Ballmer was not invited to the tech summit with Zuckerberg, Jobs, Smith, and other heavyweights. Do you buy the argument that it had to do with geography? Or is the geography or the symbolism of the Silicon Valley really so important that you would ignore a Dow Jones company that's two states away?"

Carl Franklin: Come on, everybody knows he was getting his teeth filled.

Richard Campbell: Nice. "Carl had previously mentioned that Microsoft is not primarily known for innovation the way that Google App and Facebook are. But even if their strong point is improving on innovations of others using their vast resources, wouldn't it make sense to include them?"

Carl Franklin: Yeah.

Richard Campbell: You know, I thought about this question, Mike, and you know what hit me? You know who wasn't there that you really can't justify from geography? Larry Ellison.

Carl Franklin: Okay.

Richard Campbell: Larry Ellison is right in the Silicon Valley, Oracle is right there.

Carl Franklin: Is he still around?

Richard Campbell: They own Java, you know they bought Sun.

Carl Franklin: Yeah.

Richard Campbell: They have all of this stuff. They weren't invited either.

Carl Franklin: Right.

Richard Campbell: So I don't know and honestly I wonder about how tricky it is to get all those people together at the same time too although I can imagine when the President calls you pretty much drop everything.

Carl Franklin: Pretty much.

Richard Campbell: But I'm not going to read too much into it.

Carl Franklin: Yeah, nor should you. That's just the way it goes. I didn't get invited either so come on.

Richard Campbell: Yeah. We didn't get invited. They'd asked us nothing.

Carl Franklin: No.

Richard Campbell: But, Mike, thanks so much for your email and I'll send you out a mug right away. If you've got questions, concerns, or just want to know what we think about something, send us an email, dotnetrocks@franklins.net.

Carl Franklin: Hey, Richard, you know we have a new website.

Richard Campbell: Yes, we do.

Carl Franklin: We've talked about it a little bit but it's really been up now for a few weeks and it stood the test of time as it were, of geek time. One of the things that we have on there is the Discuss Framework that Scott Hanselman has on Hanselminutes, also a link to Facebook and Twitter. So essentially we were encouraging you to leave your comments. Forget about sending us your emails about things that you like or didn't like about the show. I mean you can do that too, but we would prefer if you shared it with the world right there on the .NET Rocks website page. Just click the comments icon and go ahead and add a comment and you can link it to your Facebook and Twitter account too. Also I want to mention the big orange banana in the website, personalized RSS. So if you click on that, we started tagging our shows and you can select. You can create your own custom feed that's based on the tags. So if you want a feed that only has stuff about F# or Kinect or Silverlight or C# or whatever it is, HTML 5.0, just check off the tags that you want, say Create MY Feed, boom, you've got a URL, put that right into iTunes or whatever it is you want or your own website if you just want to follow what we are talking about in a particular topic. There you go. It's a personalized feed. We have a Tag Cloud at the top. You can click the all tags button, it shows the most popular tags at the top of the show. But there you go.

Richard Campbell: There you go.

Carl Franklin: I think it's worked out really, really well. Of course the RSS tag, if you just want all the shows, it's big. If you just click the RSS button now, you get all of the shows and not just 20 shows. We also have links to the Zune marketplace to iTunes. We have a fan page on Facebook, you can click that or follow us on Twitter. We're just all sorts of online now.

Richard Campbell: We're doing our best to connect up with you any way you want to talk.

Carl Franklin: That's right. So Richard, let's roll the interview that we did on Saturday at the New York City Code Camp.

Richard, our guest today is Gael Fraiteur. Gael is the founder of SharpCrafters and one of the creators of PostSharp, the leading aspect-oriented framework for .NET technologies. Gael's passion for programming goes back 20 years. He's a frequent speaker at user groups and conferences. Gael is Belgian and resides in the Czech Republic. He's married and has 3 kids. Welcome Gael.

Gael Fraiteur: Hi.

Carl Franklin: We are actually at the New York City Code Camp, Gael and I, in two different rooms separated by a wall calling back to Canada.

Gael Fraiteur: Yes. Don't you think it's crazy?

Richard Campbell: Don't you love the technology we use for this stuff. It's hilarious.

Carl Franklin: It is hilarious.

Gael Fraiteur: That's crazy, you know. I've traveled 6,000 miles to record this stuff and we end up in different rooms.

Richard Campbell: Yeah.

Gael Fraiteur: Synchronizing by phone. Nostalgic.

Carl Franklin: Just to get a good clean audio signal.

Richard Campbell: Yeah. And talk to a guy on the other side of the continent as well.

Carl Franklin: Yeah.

Richard Campbell: So Gael, how is Aspect Orientation going these days for you?

Gael Fraiteur: Aspect Orientation is going well. We've convinced many companies that it was worth their time, and investment, and attention. So the last time we recorded it was in November 2007 so it was the very beginning of the success of Orchard. I think after the recording you can see it in Google and then I think at this time it started to take off slowly so I think I have some debt to you guys because there's a correlation between this and .NET Rocks! So last year we decided to make the product commercial because you know I just wanted more support, more features and I could not just do it in my free time. So we started in March of last year and got a good bunch of first customers. So it's going well and an entry at the CodeGen Conference that I'm taking an avid interest. So I'm very happy with that.

Richard Campbell: Cool.

Carl Franklin: Now Aspect-Oriented programming, because we haven't talked about this in a while, I would say I don't know how popular it is so I can't say one way of the other. But the whole idea here is to remove stuff from your code from the code that you're working on so that you don't ever have to see it. Is the idea then that you end up with cleaner code, more testable code, or does it just make it easier to find the code that you're looking for when you're bouncing around from module to module, or all of those things?

Gael Fraiteur: So the idea is that there's code that you don't want to see and it is there because you need it. So what you want to see is the business code, the code doing real business operations. But isn't that enough to make the application work in foundation conditions? So you would have to add some extra handling tracing security codes, trace incarnations, locking and so on, and this code is actually different. It contributes to the business functionalities of the application but if you don't have it it won't work. So say for instance that there is am issue in production and you need to trace every method of your application and record the value of the parameters' in production because you are not able to reproduce the issued specific integer. So with Aspect-Oriented Programming you can create an aspect.

Carl Franklin: So hold on a second. What is an aspect? You say you create an aspect, what is that exactly?

Gael Fraiteur: So an aspect is a class that has tried the transformation of your code to what the aspect would contain. In case of tracing, if you want to trace before the execution of the method, if you want to trace the value of every parameter, the aspect would be a class iterating from a class on method boundary with one method on entry and inside this from entry you would have trace synergy.

Carl Franklin: I see.

Gael Fraiteur: So an aspect is a class of behavior you want to add to methods or classes.

Carl Franklin: I see.

Gael Fraiteur: So first you do the aspect in one class necessarily easy and then you apply the aspect to target. So once you had the tracing aspects, you can add it to these applications or complete namespaces in just one line of code.

Carl Franklin: So is that one line of code in the constructor of a class?

Gael Fraiteur: No. It would be for instance in your Assemblyinfo.cs. You would say that you apply the aspect to the assembly so you don't have to add...

Richard Campbell: So you're not actually modifying the code of the application itself?

Gael Fraiteur: So what the AOP framework does is that it will modify not the source code but the output of the compiler.

Richard Campbell: I see.

Gael Fraiteur: This is how a post compiler framework is done.

Carl Franklin: Hence the name PostSharp.

Gael Fraiteur: Oh, yes. PostSharp is, because this is a post compiler indeed so it happens after the compiler takes the output of the compiler, transforms it and write it back.

Richard Campbell: Right. But this sounds like something you would only use for tracing and logging.

Gael Fraiteur: It happens to be the Hello World example. So it's the first thing that we show. So yes, the first thing. It's the only feature that's useful for whatever it's useful. But you can use this, for instance, for implementation of INotifyPropertyChanged. So in WPF we have this data binding, a great feature.

Carl Franklin: Yeah.

Greg Hughes: You just say that the text property of the textbox is bound to the name property of the current context. Again, you don't have to write the glue but you still have to implement the interface INotifyPropertyChanged.

Carl Franklin: And you have to implement it in every property that you're binding and if you forget, guess what? It doesn't work.

Gael Fraiteur: Yes. So it is just three lines of code, okay. But three lines of code in every property.

Carl Franklin: In every setter.

Gael Fraiteur: So in every setter. So an aspect for this would look like let us a behavior to every property setter, and this behavior would do what? It would look at the current value of the property, compare it to the new value and if it is different raise the event property change so that in the aspect...

Carl Franklin: It's brilliant.

Gael Fraiteur: You don't have to modify every property setter. So what you would do is put the aspect on the base class of your entity model or business model and say that it is inherited and you never have to think about implementing the INotifyPropertyChanged again.

Carl Franklin: So is there a danger, Gael, because I love this idea and I would imagine myself going nuts with it, is there a wall that you can hit or can you go too crazy with this so that your code becomes out of control and starts doing... I mean, that seems to me the danger that I use it for so much that if you have a bug in your aspect for example it could manifest itself in nasty ways.

Gael Fraiteur: So if you have a bug in the aspect and actually it's much safer than having a bug in the implementation of the manual lines, okay, but if you have a bug in the aspect you correct the bug once.

Carl Franklin: Yeah.

Gael Fraiteur: If you change your mind and you don't want to implement INotifyPropertyChanged this way but another way, then you have to correct the same bug hundreds of times.

Carl Franklin: You're so right.

Gael Fraiteur: That said, you okay any Sharp tool and with any Sharp tool you should pay attention. You can do wrong things if you want and the framework would try to prevent you doing that. There is some kind of guards and checks. So AOP is a good approach to programming.

Carl Franklin: Your point is definitely well taken.

Richard Campbell: All I was thinking was don't run with scissors.

Carl Franklin: Yeah, exactly. That's why they're sharp.

Richard Campbell: This stuff is sharp. You can cause problems. But you know, all good tools can hurt you.

Carl Franklin: Of course. I mean, in C# you can do that too. So the tool PostSharp, does it not only include the framework but lots and lots of different aspects?

Gael Fraiteur: So the framework, it may contain aspect that you can use out of the box. It is a framework that allows you to create the aspect. So it gives you the blocks and then you have to implement the aspect.

Carl Franklin: But I imagine once the aspects are created they can be reused quite a bit.

Gael Fraiteur: These aspects are given as samples. So you take the sample code and modify them. For instance INotifyPropertyChanged, it has 50 lines of code and probably in every application it is implemented a little differently in different kinds of -- just don't work similar and so on. So we give the source code of this aspect and you can start from this. The PostSharp, itself doesn't provide you with aspect. It provides you with a toolkit that lets you build aspect.

Carl Franklin: So it sounds like these aren't aspect like custom controls that you can just drop on and point to your assemblies and magic happens. You really do have to think about how they fit into your application.

Gael Fraiteur: Yes. But typically if you have a large team or even a large department of investment, one of the teams is responsible to the needs of the architecture. It can select the stack of applications and frameworks that will be use in the architecture. They create the -- they did this framework so they would say we would use NHibernate, we would use Castle, and we would use PostSharp and they would create the business framework, the base classes and so on, they would create design patterns and deliver to the line of business developers or the UI developers and say it is the way we would like you to code as a team. So normally the architecture guys would do the aspect, do the low level stuff and play whistle locks and so on and deliver to the team readymade aspects that work just out of the box. This is how you can really separate the concern of everybody. The architecture team is responsible for the technical side of the software, and the business team do the business plus user interface as one.

Richard Campbell: Now when we talked to you in 2007, PostSharp was just an open source project, wasn't it?

Gael Fraiteur: Yes, it was. It was an open source project and there's 1.0 or 1.5, I don't remember at this stage. Now we did a new version, it's much more robust, and the new version is available as a commercial product.

Richard Campbell: Okay. And there's still a community edition available.

Gael Fraiteur: Exactly. So most of the features of the former open source edition are in the free edition.

Richard Campbell: Okay. So why make a retail version?

Gael Fraiteur: Well, because you have to make a living of this.

Carl Franklin: That's the best answer I've ever heard. You have a free version. I'm sorry, that's funny.

Richard Campbell: Yeah. A free version was fun but I couldn't eat.

Carl Franklin: Sorry, Gael.

Richard Campbell: But also, I mean folks obviously are using this and wanted more and more from it, and if you're going to work that hard on it you've got to be able to pay your bills.

Gael Fraiteur: That's exactly right. The first step i took into 2.7 was some sponsored halftime and do halftime on level three and each work doing one year, two years and then I figured out there's no way to continue this way.

Richard Campbell: Right.

Gael Fraiteur: And it's not just me. The users told me repeatedly we want a commercial version, we want to be able to pay you for your work. But you know we are a commercial company. If we are not forced to pay we will not pay you because it's how business works. Do a commercial version and we will buy it.

Richard Campbell: Right.

Gael Fraiteur: Many users told me this so I don't mind. So we went for it.

Carl Franklin: This portion of .NET Rocks! is brought to you by our friends at Telerik. So you know all about the power of ASP.NET MVC, but you might be in need of some good tools to enhance your productivity. Well, our friends at Telerik just shipped the latest release of the Telerik extensions for ASP.NET MVC, 18 jQuery-based native MVC extensions. Now you can enhance productivity by remaining in control of your views without having to write all HTML, CSS, and JavaScript by hand. Did I mention that the Telerik MVC extensions are also free and open source? Plus now you can check all MVC online demos in both ASPX and Razor View since the extension offers full support for ASP.NET MVC 3.0 and the Razor View Engine. Download your free copy today at www.telerik.com/freemvc, and don't forget to thank them for supporting .NET Rocks!

I'd like to talk a little bit about multithreading aspect because I'm reading the PostSharp definition on your website and you mention that multithreading is a good application for aspects. How would that look?

Gael Fraiteur: So in multithreading, that means that you can deal with aspect. The first is putting a method on the thread. For instance if you are doing the user interface, you are doing data accesses then you know that you cannot do I/O or to slow I/O on the user interface thread because if you're doing that you're blocking the method's queue and the user interface's threading so you have to do I/O and iterate aspect on the multithread.

Carl Franklin: Yeah.

Gael Fraiteur: So what you could do in PostSharp, instead of calling threadpool and you work out a demo, whatever, you could do an aspect like on work a thread attribute and then you could discuss the attribute on the method that needs to work asynchronously and you can do operative things when someone needs to work on the GUI thread. Because they're a bit the user interface, you would put an attribute that dispatches the method to the user interface thread. So just by annotating the method with the right system attribute, you can define on which thread every make are different and it can clean up the code the right way and, you know, C# 5.0 is going to make this much easier to work with multithreading.

Carl Franklin: Right.

Gael Fraiteur: But the point here is that this feature has been available for four years or five years. We're not waiting for C# 5.0. You can do this today, you could do this yesterday.

Carl Franklin: Right.

Gael Fraiteur: The second part is that once you have multiple threads, you need to synchronize access to share and to enforce it. That means using locks. One of the ways to enforce deadlocks is to use either reader-writer locks. So when you need to write access to an object, you take a write lock and when you need to just read it you take a read lock. You can have multiple read locks concurrently, but you cannot have a write lock countering with a read lock. So if you have to play with this primitive called ReaderWriterLockSlim.a car reader lock and so on, it's going to upload to your code. What you can do instead of that is to make an aspect. For instance Read Lock Attributes and then you annotate the methods of your code with Read Lock, Write Lock and so on.

Carl Franklin: So yeah, I think what you say about C# 5.0 is a good point, that a lot of the stuff is going to be obsolete. But if you are doing locking, I think I would prefer to know where my locking is. I mean, hiding threading code is scary to me and I need you to allay my fears.

Gael Fraiteur: Oh, but it's, you know, you don't hide this. First in this case, in case of thread locking, you would not use one car to say I need a Read Lock on all the methods of this namespace. It makes no sense for this kind of aspect. What you would do is to add a customer if you do every method requiring a lock. So what you have is that instead of having code that does it, well, you have a customer that you greet on your methods so the code is more readable and you don't lose the information. You have a lock. You still know from reading that they see that there is a lock but they don't see the implementation.

Carl Franklin: And I think the point there is that this is code that you probably wouldn't write yourself because you have to write it on every method, but when you do have code that is consistently written on every method you can do things that you wouldn't normally think of.

Gael Fraiteur: Yeah. What's new in PostSharp too is that inside Visual Studio when some method is enhanced by an aspect it will likely own a line inside the code editor so that you see just by reading the code and seeing the code you see, oh, there's an aspect on this method, then you move the cursor and the tool lets you see the list of old aspect that has been applied. So this is a way to -- we could take the issue that you're not sure which aspect or where, and actually it's very likely that your colleague added an aspect to your code and it's okay, it's okay if you are able to see that. So that's one of the most important features in .NET and PostSharp too, it's its IDE support. So I've seen that this fear of the code changing behind is so under the hood without any control. So there is a part that is a little irrational. There is a part that is justified and for this part we offer the tools to increase visibility and to add and increase the sensibility of the code.

Carl Franklin: Do you ever get feedback from customers telling you or showing you code that they've written with aspects that you didn't think of doing and you said whoa, that's an interesting application?

Gael Fraiteur: Yes. I really did have a customer and I've been very interested to see a large scale undo, redo aspect because they're doing a 3-D editing software so they have a complex object model with hundreds of objects and thousands of properties. The undo, redo aspect would actually recourse every change to every feed of the domain.

Carl Franklin: Wow.

Gael Fraiteur: The aspect is just some lines of code.

Carl Franklin: Undo and redo, that's brilliant.

Gael Fraiteur: Yes. What is undo, redo? You just have to recall that this feed had this value and now has a new value. What is undo? Undo is assigning the old value, and redo is assigning the new value.

Carl Franklin: That's great.

Gael Fraiteur: So that's easy. What's hard is to recall everything and this is made super easy with AOP.

Richard Campbell: That's my favorite example so far of Aspect-Oriented Programming.

Gael Fraiteur: Yeah. I agree, Richard.

Richard Campbell: Plus you could do stuff like actually save the undo, redo stacks separately so you could even reload an object or a document and go back and replay it, even look if there's a log like you've just taken this whole thing to another level.

Gael Fraiteur: Transaction log like restart the application. So it was very nice to see that, and what's also nice to see is that I see they have problems with .NET 1.5 and I anticipated these problem and there was already a solution in version 2.0 even if I didn't see the problem in real code. So for instance, there were issues that undo, redo aspect would interfere with the INotifiedPropertyChanged aspect and so on and these issues were solved in PostSharp 2.0. So I was very happy to see this code in very great application.

Richard Campbell: So I think the only real problem with PostSharp is that Anders Hejlsberg doesn't like it.

Carl Franklin: Now why is that?

Gael Fraiteur: Maybe.

Richard Campbell: Well, I saw the video of Anders. Was it PDC '10?

Gael Fraiteur: Yes.

Richard Campbell: Where he was fairly negative about Aspect-Oriented Programming as a whole. I don't think he's speaking on PostSharp per se, but he did not want to put AOP into C#.

Carl Franklin: Well, that's good for PostSharp I guess.

Gael Fraiteur: But you know there are two things. You're poking AOP is one thing and then inserting AOP into C# is another thing. So there will be direct support of the AOP inside C#, I know directly. I would not answer for the delay. I think it would be like a big open door. AOP is a huge stack. There's lot semantics to the language. I don't think that the language is the right way to implement this.

Richard Campbell: Right.

Gael Fraiteur: It's still got what happened in the Java world, they started the Project Aztec J and it started as a language extension. You find a new keyword and now the current version of Aztec J are not implemented using new keywords but using annotations because remember that in 2001 to 2003 Java did not have annotations. .NET has. Through the annotations we could have AOP. In .NET we don't support the language. So this question should the language support AOP, I would answer no, it's not necessary. I think the language should stay distinct, it should stay very minimal, as minimal as possible. We don't want C# to be compared. Quite surprising that can be done to make the C port easier and to improve experience. So there are two things, do porting in the language and I perfectly agree with the provision that it should not be integrated in the language, but then being completely opposed to AOP by transfer I think this is where I don't agree anymore. I think that there is a point for AOP especially in these applications and there are many, many customers that want AOP and it's not just hard to look at. Look at ASP.NET MVC, they have the action syntax.

Richard Campbell: Right.

Gael Fraiteur: What is action syntax now in AOP? Look at WPF, they have got some behaviors.

Carl Franklin: Yup.

Gael Fraiteur: That's a form of AOP. So many frameworks need AOP and implement it in a partial way because they don't have the right technology. So if there is a need for AOP, it has been recognized in many, many projects outside the ASP.NET MVC, WCF, Unity, Dependency Injection, Application Block. These are just Microsoft products.

Richard Campbell: Gael, a lot of these sounds like stuff that Dependency Injection could do as well.

Gael Fraiteur: Well, Dependency Injection is great. It works with AOP if you are happy to add your behaviors just in the boundaries of your service.

Richard Campbell: Right.

Gael Fraiteur: So supposed you have a client in a service, your client calls a service, it causes a boundary. If at this point you want to add an aspect, that's great. But what if you need to add an aspect inside of service because the aspect is on the domain object process, then what do we do?

Richard Campbell: Right.

Gael Fraiteur: So you cannot choose. You can't choose Dependency Injection, or what would happen is that, well, the framework would say you can do AOP with any object but you need to make every property virtual for instance or any method virtual because it is their job and we are able to actual the transact. This is what happens in NHibernate. They tell you we support Lazy Loading which is an aspect. If you support Lazy Loading, please do all your properties virtual. So what's happening there is that you're changing the architecture of your code simply to offer a hook to the AOP Framework or to the Dependency Injection framework. In some cases, you are creating simple a team of your own architecture because you are optimizing for the requirements of the Dependency Injection Framework and you start choosing factory method and so on. So anyway, what if you cannot choose a factory method? For instance, what if you want to add a thread synchronization aspect to a method in WPF or Win Forms? You cannot create a control with a factory method and add it from the container.

Richard Campbell: Okay.

Carl Franklin: Hey, I just want to give a shout out real quick to our friends at Data Dynamics who make ActiveReports.NET among other awesome things. ActiveReports.NET is great because it allows you to just build your reports with the Easy Editor, embed them right in your application, provide PDF and HTML output, give your end-users a Report Editor, royalty free of course, a great Access report upsizing Wizard and all this for a price that isn't going to break the bank. ActiveReports.NET from Data Dynamics, go check it out now at datadynamics.com.

Gael, can I switch topics here and talk about testing? It seems to me that you have a lot less tests to do when you don't have all that goo in your code, but does it do anything to code coverage if a lot of your code is not testable shall we say?

Gael Fraiteur: So what do we want to test? We want to test the code that was written by the OS. We don't want to test the code that was written by the framework. So there are two things we need to test. We need to test the aspect separately. You make unit test for the aspect, and then we make a test for code as usual. So you don't need to test the code that has been generated by the AOP framework because this code is what makes it generated. That's okay.

Carl Franklin: But the interaction between your code and my code certainly produces a whole new thing that could, the interaction of which could fail.

Gael Fraiteur: I think the right answer here is the event. It depends if direction is orthogonal or not. Say that you have a tracing aspect that you add to database code. There is no interaction between tracing and database so you don't need to test the interaction.

Carl Franklin: True.

Gael Fraiteur: Say that you're using the transaction aspect on database code. In this case you need to test. That's clear, okay.

Carl Franklin: Right.

Gael Fraiteur: So you need to test the interaction when there's an interaction, and when it is orthogonal I think there's no need to test. You just need to test the aspect and the base code separately. This is what I would recommend. There's one I brought here that some test coverage tools are not smart enough to make a difference between the show that has been generated manually and the code -- sorry, the code the has been generated automatically and the manual code.

Carl Franklin: Yes.

Gael Fraiteur: So it would report a method coverage of 50 persons because a 50-person public code is automatically generated.

Carl Franklin: Yeah.

Gael Fraiteur: So this has been reported. I think it's an issue of the code coverage tool because PostSharp, it needs all the TDD information to make clear that this is automatically generated code and doesn't need to be the target of a breakpoint either by the developer or code coverage and so on.

Carl Franklin: Do we know anything about end unit or the testing tools in the TFS queue?

Gael Fraiteur: I don't know to which two certificates it is. This is like report quest as related to. But you know, it's the same issue. Say that you make an iterator in C#. Was it two or three trees? But probably you can make iterators. The compiler would generate more code that you actually wrote. Through this code, the coverage -- or if it's a generated code that, then that test coverage tool would be smart enough not to require the coverage of this code.

Carl Franklin: I see.

Richard Campbell: Gael, I've heard these terms but I don't necessarily understand the difference so maybe you can explain it to me. Static versus dynamic AOP.

Gael Fraiteur: So static AOP happens at the bin time. Static AOP is PostSharp and in Java as the J is mostly static AOP. So we change the code of your application, not the source code but the bind code, the MSIL code. Dynamic AOP, it happens at runtime so it means that most of the time the Dependency Injection framework would add behaviors at runtime and you don't recompile your application or change any. So this works well with the Dependency Injection concept that when you want an object you don't choose a new keyword in the constrictor but you ask the container please give me an implementation of this interface.

Richard Campbell: Right.

Gael Fraiteur: The IoC container, if there's an aspect on the service, what it would do is it will not give you the service. It would give you instead a proxy that is generated using Namespace: System.Relfection.Image. This proxy would implement the interface that you require, but instead of directly calling the service that implements interface it would call the aspect before and after or instead.

Richard Campbell: Okay.

Gael Fraiteur: So you can add behaviors, thanks to the proxy but it will not change a service. It is all runtime work. So runtime is great when first you use Dependency Injection, you want aspect on the boundaries of the service because you can't change the aspect result through recompiling the application. Say that your application is in production. You're having trouble, you cannot distribute it, you want to trace all the furniture of your internal service that is exposed to a tester friend. So let's say Spring.NET for example because I know this one better. So what you would do with Spring.NET is you would go to the application configuration file and you would configure this service and you would say that this aspect, the threading aspect should be applied to this service. So you restart the application and you have the aspect applied. With the static reserve you will need to rebuild. So the static reserve are very powerful because you can change everything in the application. You can do much richer transformations. There are compiled validations and many useful things, but once it is compiled it is frozen. What you see using the compiler is what gets executed with the runtime, it all happens at runtime and you can change it without recompiler.

Richard Campbell: That's cool.

Carl Franklin: Nice.

Richard Campbell: Yeah, that's very clever. It's almost a plugin model.

Gael Fraiteur: The plugin model is what?

Richard Campbell: It feels like you could do plugins that way.

Gael Fraiteur: Well, in the runtime situation, yes, the aspect works as plugins.

Richard Campbell: Right.

Gael Fraiteur: Yes. Because you can do aspect and separate assemblies and you can add them to the pipeline. This is exactly what you need that it does and you can configure exactly. So actually PostSharp is also like in oriented, but the plugin system occurs at bin time and not at runtime.

Richard Campbell: Okay.

Carl Franklin: So Gael, what's next for PostSharp? What do you have on your drawing board?

Gael Fraiteur: So we would like to make PostSharp yet easier for the users. So we would like to make a user interface so that it can add aspect just using your Wizard and so select the method or use rules but using your user interface. I think that many of the few people we are reaching now, they don't want to read documentation. They just want to click and see the benefit, and if they see the benefit they would go with the learning curve.

Carl Franklin: Right.

Gael Fraiteur: So make it easier on one side, on the other side there are some aspects that we would like to deliver as a box. So currently what we have to say is that we provide you with Lego bricks and you can do a car with it, you can do a plane with it, but you have to do it yourself.

Carl Franklin: Right.

Gael Fraiteur: Now what we also say is just have the Lego bricks but we also provide a plane and a car if you need it.

Carl Franklin: Yeah, that's sort of what I was getting at before when I thought, wow, it would be cool if there was a bunch of aspects that were just baked-in already that I could just attach to my assemblies.

Gael Fraiteur: Exactly. So this if for medium term, not a short term so it's happening now. What we are doing now is improved the experience of current users and here, I mean, improve compile time and to correct some of the dark spots into using 2.0. So version 2.1 is basically processing the backlog, fixing things that need to be fixed. Version 3.0 will be the out of the box aspects and more usability.

Carl Franklin: Fantastic. And it's postsharp.com, that's where we can find this?

Gael Fraiteur: It's sharpcrafters.com. I have an offer for .NET Rocks! listeners. We are going to give away five free licenses amongst those who signed up within 72 hours following the publication of this site.

Carl Franklin: Nice.

Gael Fraiteur: And to sign up, go to http://bit.ly/postsharprocks.

Carl Franklin: Okay. And we'll also post a link to that on our webpage, dotnetrocks.com, in case you miss that, and 72 hours.

Gael Fraiteur: 72 hours following the publication and we will take five, so you don't have to worry. You don't have to hurry up if you miss the episode by one hour. You're going to have three days.

Carl Franklin: All right, great. Thank you, Gael. This sounds great.

Gael Fraiteur: Thanks.

Carl Franklin: All right. Thanks for talking to us, and thank you, listeners. We'll see you next time on .NET Rocks!

[Music]

Carl Franklin: .NET Rocks! is recorded and produced by PWOP Productions, providing professional audio, audio mastering, video, post production, and podcasting services, online at www.pwop.com. .NET Rocks! is a production of Franklins.Net, training developers to work smarter and offering custom onsite classes in Microsoft development technology with expert developers, online at www.franklins.net. For more .NET Rocks! episodes and to subscribe to the podcast feeds, go to our website at www.dotnetrocks.com.