



[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 #495*  
(Transcription services provided by [PWOP Productions](#))



**Vittorio Bertocci authenticates us with WIF**  
**November 3, 2009**

*Our Sponsors*



**Developer**  
EXPRESS

[HTTP://www.devexpress.com](http://www.devexpress.com)



**CoDe**  
component developer magazine  
[HTTP://www.code-magazine.com](http://www.code-magazine.com)



**RadControls**  
FOR ASP.NET  
telerik  
[HTTP://www.telerik.com/](http://www.telerik.com/)



**Geoff Maciolek:** The opinions and viewpoints expressed in .NET Rocks! are not necessarily those of its sponsors, or of Microsoft Corporation, its partners, or employees. .NET Rocks! is a production of Franklins.NET, which is solely responsible for its content. Franklins.NET - Training Developers to Work Smarter.

[Music]

**Lawrence Ryan:** Hey, Rock heads! Stop cracking your tokens and listen up! It's time for another stellar episode of .NET Rocks! the Internet audio talk show for .NET developers, with Carl Franklin and Richard Campbell. This is Lawrence Ryan announcing show #495, with guest Vittorio Bertocci, recorded live Saturday, October 24, 2009. .NET Rocks! is brought to you by Franklins.NET - Training Developers to Work Smarter and now offering SharePoint 2007 video training with Sahil Malik on DVD, dnrTV style, order your copy now at [www.franklins.net](http://www.franklins.net). Support is also provided by Telerik, combining the best in Windows Forms and ASP.NET controls with first class customer service, online at [www.telerik.com](http://www.telerik.com), and by CoDe Magazine, the leading independent magazine for .NET developers, online at [www.code-magazine.com](http://www.code-magazine.com). And now, the man who says, "Two's company, three's a scrum meeting," Carl Franklin.

**Carl Franklin:** Thank you very much and welcome back to .NET Rocks! It's a beautiful fall day here in the United States. Richard and I are just back from our whirlwind tour of Europe and we're on to Sweden, Las Vegas, and Los Angeles to PDC. Hey, Richard, what's up?

**Richard Campbell:** I'm battling the same cold as you are, sir. I think we should all thank Shawn Wildermuth for infecting the entire speaker core that went on that European tour.

**Carl Franklin:** Well, who knows if he infected us all but he certainly got it first.

**Richard Campbell:** Yeah, that's true.

**Carl Franklin:** And it's nothing to worry about, it's not SARS and it's not H1N1. It's just a little head cold.

**Richard Campbell:** It's a cold.

**Carl Franklin:** You pick it up. Hey, better than last time I went to Bulgaria. You remember that?

**Richard Campbell:** Yes, I do actually.

**Carl Franklin:** I had to go to the doctor after that.

**Richard Campbell:** Yeah, no kidding.

**Carl Franklin:** I don't know what that was.

**Richard Campbell:** It was something spectacular.

**Carl Franklin:** It was.

**Richard Campbell:** But I've also discovered that three conferences in two weeks and a cold actually can drive me out of a bar.

**Carl Franklin:** Yeah, that's true. The last two days in Amsterdam there, we were just dragging.

**Richard Campbell:** We were going to bed by 10:00 like good little kids.

**Carl Franklin:** Although there is a video of me in Amsterdam somewhere playing "Ain't No Sunshine When She's Gone" to an Egyptian, a Dutch guy, you and Stephen Forte.

**Richard Campbell:** It's awesome.

**Carl Franklin:** It was a thrilling adventure. Well, anyway, let's get right into Better Know a Framework and start this puppy off right.

[Music]

**Richard Campbell:** All right.

**Carl Franklin:** You're going to like this, Richard.

**Richard Campbell:** Yeah.

**Carl Franklin:** Have you ever had a multithreaded application where you're creating your own threads or where you're using the asynchronous model or whatever it is, but you have a thread and you wish you could stick some data on that thread that only is accessible on that thread.

**Richard Campbell:** Right, definitely.

**Carl Franklin:** Yeah. Well, you can do that with the ThreadStaticAttribute. The System.ThreadStaticAttribute allows you to -- you put this on a static field...

**Richard Campbell:** Oh, yeah.

**Carl Franklin:** And then you can access it differently for each thread. In other words, every thread gets its own copy of that field.

**Richard Campbell:** Cool.

**Carl Franklin:** Yeah, that's it.

**Richard Campbell:** So actually I can push this to multiple threads and they'd all be isolated from each other.

**Carl Franklin:** Absolutely. So the idea is that -- and it doesn't matter, every single line of code executes on a thread. So even if you're not doing a multithreaded application, you have a thread and if you use System.Threading. -- what is this? Thread.CurrentThread, or Current I think it is, it returns the thread that you're code is operating on.

**Richard Campbell:** Right.

**Carl Franklin:** Within that, if your class that you're currently executing in has a static method with the ThreadStaticAttribute on it, you can get and set the value of that within that thread only. Think about that.

**Richard Campbell:** It's awesome.

**Carl Franklin:** Yeah, that is awesome. Maybe you have a use for it. I can smell the gears of the listeners turning.

**Richard Campbell:** Oh no, I'm definitely thinking about it. And of course at the same time I'm also thinking about all the conversations we had recently about how threading is going to be done going forward with new libraries and that in theory where if -- you know, one of our guest said "Hey, if you actually say go spin up a thread, you've done something wrong."

**Carl Franklin:** Yeah, that's right. I don't necessarily agree with that because sometimes that's a simple solution to a simple problem, but I think the point is there that there's so many other technologies now for doing parallelism that you really ought to look into those before you try to roll your own.

**Richard Campbell:** Absolutely, and I also think that the real statement here is there's multithreading and there's multithreading. The idea that you're going to fire off a background thread to do something, that's what this class is for and it's a good practice if you have an app that makes sense on. But this massive parallel environment that's coming, you don't want to spin up your own threads for that.

**Carl Franklin:** No, no. Hey, who is yakking at us now, Richard?

**Richard Campbell:** I have an email from actually a good acquaintance of ours who we've bumped into many times, Robert Cain...

**Carl Franklin:** Oh yeah.

**Richard Campbell:** Whose pseudonym is Arcane Code.

**Carl Franklin:** Right.

**Richard Campbell:** Who send us an email about #483 and starts off the email so well. "Dear Dynamic Duo of .NET. I always enjoy it when you throw a little love in the SQL Server arena, and Kent Tegels' show on SQL Server Integration Services was especially good. A long time developer turned BI guy myself, like Kent I've been evangelizing the use of SSIS in the developer world through presentations at various code camps and .NET user groups. There is one point though that I don't feel was made entirely clear in the episode. It is indeed possible for a .NET application to run an SSIS package if that SSIS package has been deployed to the server. You would set up a SQL job within which to run the package then call the SP Start Job store procedure from your code. I created this sample that uses .NET to upload a list of previous DNR episodes then invokes an SSIS to parse them and to show a number, name, and dates in a new table..."

**Carl Franklin:** Sweet.

**Richard Campbell:** "And you can find all the instructions at [shrinkster.com/1a7i](http://shrinkster.com/1a7i)." That's 1, Alpha, 7, Indigo.

**Carl Franklin:** Now there is an alert listener.

**Richard Campbell:** No kidding. "Keep up the great shows and keep mixing in a little SQL once in a while. And hey, tell Paul and Kim the honeymoon is over, time to get their podcast off the ground."

**Carl Franklin:** Yeah. Come on, let's do it, guys.

**Richard Campbell:** Let's do it. "Thanks from Robert Cain." Robert, dude, I don't think you have a mug, but you have one coming now. And if you have any questions, concerns, ideas for shows, comments or criticisms, send us an email, [dotnetrocks@franklins.net](mailto:dotnetrocks@franklins.net).

**Carl Franklin:** And with that let's introduce today's guest, Richard, Vittorio Bertocci. Vittorio is a Senior Architect Evangelist in the Azure Evangelism team with Microsoft. After a few years in the Italian Microsoft Services, he moved to the U.S. headquarters where he has spent the past four years helping customers deploy solutions based on identity and access management, SOA, and services. He currently focuses on all things Identity, working with



the developer community, large enterprises and partners. Vittorio frequently speaks about Identity at international conferences and maintains a popular blog at [blogs.msdn.com/vbertocci](http://blogs.msdn.com/vbertocci). That's two C's. "Understanding Windows CardSpace", the book on user-centered identity he coauthored was published in January 2008. Welcome, Vittorio.

**Vittorio Bertocci:** Bonjour everybody, thank you.

**Carl Franklin:** Bonjour.

**Richard Campbell:** You know, you don't hear much about CardSpace these days. Has it morphed?

**Vittorio Bertocci:** Hey, CardSpace is currently being integrated in, I thought you may have heard, the Geneva wave in which way it is really shining far in the enterprise scenarios. So CardSpace is definitely there, it's much, much improved. It's very fast, super agile, it's very, very usable and it's very well integrated with the enterprise scenarios. So for example, now you can distribute CardSpace with group policies and similar. So yeah, CardSpace is still very much alive and kicking.

**Richard Campbell:** But very much an enterprise technology. We're not seeing the consumers pick it up much even though it's shipped with Windows.

**Vittorio Bertocci:** Right and now it's, I believe, designed to be integrated with MSN of course and there's a survey there. They can see the end-user part, but it's namely in -- the Geneva part -- is really fine for enterprise scenarios.

**Carl Franklin:** Well, we spoke with Kim Cameron back in, what? 2008 I think, and then Barry Dorans again on OpenID and CardSpace and I've always been impressed with the thoroughness and thoughtfulness of the architecture. But I think, as Richard mentioned, getting people to use OpenID and CardSpace for ordinary online retail, you know, Amazon, Ebay, that kind of stuff, I don't think that's where this really is focused. Are we right about that?

**Vittorio Bertocci:** I think that we will get there. That's to say that it's quite a big step because we're starting from the mess that we have today over different protocols and of people really who are not trained to think about their identities, what it means like not a lot of people think beyond the username and password. Our identity is the attribute that defines us and not just that. And as we go forward, I believe that we will see more and more update of this both because of the need of having something that is more usable like literally you cannot remember all your credentials, and at the same time because the level of sophistication is going up. However, that's a

process. You can't expect everybody to jump the shark in one single move

[Laughter]

and especially with CardSpace. We have this, I have to say it's a Catch-22 in which we needed the client with all the various capabilities but then of course you also need the capabilities for the server-side. So you need it to enable developers to write websites that will accept the security tokens coming down from CardSpace or from any other protocol, and then you also need to keep the identity providers, and that's to say the entities that know about the users, to expose that knowledge through standards and that all means that's all the nasty, low level infrastructural code if you have to write things yourself. In fact, when we came out with CardSpace, one of the examples that we had there were of a low level WCF codes and of course not everybody is a security expert and not everybody wants to write infrastructural code. Instead now that we finally have Windows Identity Foundation for developers and ADFS 2.0 for identity providers and enterprises in general, we now finally have the means to enable all this knowledge and all these capabilities to be unlock and use in a simple fashion. So I believe from now on we should start looking at what happens in this space because before we simply gave everybody -- the clients -- the server-side infrastructure. So it was kind of harder for people to actually use this technology.

**Richard Campbell:** So we've got a new foundation, Windows Foundations thing that not everybody knows about, Identity Foundation. Boy, I hate that name. But how is this different from the previous things we've released like Geneva.

**Vittorio Bertocci:** This is an excellent point. Geneva was, it got a name that kind of provided an umbrella for all the new wave of a product from the Federated Identity Team and it included the Geneva Server which then got the official name ADFS 2.0. It included the Geneva framework which then got the name Windows Identity Foundation and it included Windows CardSpace Geneva which, guess what? Became Windows CardSpace 2.0.

**Richard Campbell:** Okay.

**Vittorio Bertocci:** So they are actually the same. It just happened, like every time in Microsoft, eventually it's moved from the codename to the official name. Specifically, Windows Identity Foundation is released after they waited for years and now that they finally have I'm as happy as a kid in a candy store. This is actually an extension to what you would use in .NET for handling identity which allows you to do so in a completely platform independent fashion. It announce the developer that are not

security experts to tap into the power of the infrastructure so they can obtain attributes claims about the users without having the faintest clue about what's happening behind the scenes, and conversely it allows the security experts to take complete control of every aspect of authentication, authorization, personalization. It just decouples all the handling of the infrastructural code like for example the nitty gritty details of the protocol that you want to use or things like a checking signature, decrypting, extracting and putting it in the context of the claims from security tokens, all stuff that is Chinese for a lot of people.

**Carl Franklin:** Right.

**Vittorio Bertocci:** It actually gets made automatically. But if you want to have a hand and a say about what it's doing, you can. That's to say that now there specific places outside of your application where you can go, you can have a class, you can change a configuration, you can twist there, you can decide how long your session is going to last, and you can decide which claims you accept, you can do very sophisticated authorization instead of just saying is this your user and administrator. You can literally make decisions if the user is far from the age range. Again, all this can be done there without touching the application. Everything lives in a pipeline in front of it, and this is really the point that I believe is super important, you don't have to understand anything of what is going on if you are happy with the default which is the vast majority of the cases. Because if you're writing a UI and you just want to make sure that you accept identities from a separate identity provider and you want it to handle a certain attribute, you don't need to learn how to use the X509 API, or the Active Directory API, or you don't need even to know that you're using XAML. Those are all things that we take care of.

**Carl Franklin:** We're salivating over here.

**Vittorio Bertocci:** All right.

**Carl Franklin:** It sounds delicious. I would love to take a bite. So I'm the developer and my security guy has somehow figured out how to configure this stuff and it's probably going to take them a lot longer than it's going to take me, I can see that coming. So let's say walk us through a typical scenario in an enterprise where there's an identity provider. Let's say we want to do just some basic authentication between a couple of domains and so the security people have gone through and set up the tokens and set up the servers and all that. As a developer, walk me through what I need to do.

**Vittorio Bertocci:** Perfect. So first thing, you sit in front of your workstation and you log in. Do you want me to start from later or...?

**Carl Franklin:** No, that's fine, that's fine.

**Vittorio Bertocci:** What happens when you wake up in the morning?

**Carl Franklin:** Make it as complex as you possibly can because it's obviously going to be a piece of cake.

**Vittorio Bertocci:** All right. So it's actually incredibly straightforward. You open Visual Studio or open your existing application assuming it's an ASP.NET or a WCF application or you just start from scratch. Your choice. If you start from scratch, there are templates that are already configured for using claims-based identity. But personally, I prefer the scenario in which you take one existing application because it really shows the beauty or the simplicity of all this. You just right click on your project and you see in the menu that comes out in Visual Studio that there are a couple of new entries in your menu. One of the entries says modify STS reference. If you click on that entry you will go through one dialog that substantially just help you to pick one specific identity provider which will in practice just means making a Control C on an email that you've got from your administrator, and Control V...

**Carl Franklin:** Wait. You said modify STS? SDS? What are the letters there? What is it?

**Vittorio Bertocci:** You're right, I'm sorry. I went ahead of myself. The terminology is a typical identity jargon. STS stands for Security Token Service.

**Carl Franklin:** STS, okay.

**Vittorio Bertocci:** STS. This is a very special flavor of web service or even a stage which substantially follows some kind of protocol which could be WS-Trust, or XAML, or WS-Federation and it enables this identity provider to publish information about users on demand.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** So you use an STS for requesting a token with claims inside and you perform this request using messages that are described in those protocols. Again, for the end-developer, let me call it that, that's just literally the method of learning new words but you don't need to go into details like you just know you're going to that entry and you're just needed to provide a certain address in this wizard.

**Carl Franklin:** This is what the security people will give you in that meaning that you dread going to where you have to -- yeah, they'll give you a handout.

**Vittorio Bertocci:** Yeah. They will just give you a string, a URI.

**Carl Franklin:** Yeah.

**Vittorio Bertocci:** It will be HTTP://name of your company is the name of the kind of endpoint.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** And typically in an enterprise scenario like you've described, this would be in ADFS v2 entry. So that if you Active Directory, ADFS will do, among many other things, it puts in a standard on your domain controller. So you are able to request identities from your domain controller that are in, for example, XAML format instead of the usual KERBEROS. So it elevates the capability of your identity management infrastructure to standard base, and of course I say ADFS because I'm a developer there at Redmond, but if you'd ask any of my counterparts in other companies we actually have approved interoperability with them.

**Carl Franklin:** Active Directory Federation Services.

**Vittorio Bertocci:** So as a developer, you don't even know the platform from where those identities are coming from.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** You just get this address, you specify in this wizard, you just go ahead a couple of clicks, you see the list of claims that you will get from this user and once you hit okay your application will be automatically configured for you. That's to say that the system will go into the web.config, will add the necessary entries for making it so your app refers to an entity provider when it comes to identifying users, and from that moment on when you hit F5, if it's a web app you'll get to the browser and the browser will show for a split second your application, but then it will redirect you straight to your identity provider which you have specified with that address. Then any provider will, depending on your relationship with it, give you a token directly. Like for example, in the situation that you have described before to enterprises, when you go to your identity provider, if you are in your internet you are already authenticated because you are already immersed in your KERBEROS bus...

**Carl Franklin:** Right.

**Vittorio Bertocci:** Let's say that the data is authenticated in there, and so you will get your token silently. If instead you're going somewhere else, or

you're outside of your internet, you're luckily going to get the page where you're authenticated but it's the identity provider page. And once you've done that then you get redirected into that and you are authenticated. So just to summarize, you right click on your project, you choose this wizard modifier ADFS, you provide the address of the identity provider, you hit the play and from that moment on the authentication of your app is taken care of like you don't have to give it another thought.

**Carl Franklin:** Sounds like making a proxy for a web service. Just put in the URL, click okay, and boom, magic happens.

**Vittorio Bertocci:** You are quite correct, sir. This is in fact is very, very similar. That's to say that in the end what happens is that you're actually doing exactly what you just described that in the web service case you're waving into when you process this fact which you reach out before gathering this token, and then you use this token for calling. For the passive case, how we like to call the web browser case, what happens is that you do the same but instead of using your proxy you just have a redirect.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** Except when you do forms authentication, you just redirect it to one page in your own website which is log-in STS usually.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** In this case you just get redirected to the page of the identity provider that you trust and it's really that simple. Like you know that when we people use identity, we really like to talk about the claims idea because it is a beautiful idea so we typically like to shout it. But the reality is we've all these new tools. If you're interested in knowing more, you can certainly dig into the claims idea and similar.

**Carl Franklin:** Right.

**Vittorio Bertocci:** But if you're just interested in getting the job done because maybe you are a UI developer so you just wanted to get out of your way the authentication step, it can be as simple as those couple of clicks, you don't need to understand anything more than that.

**Carl Franklin:** Okay, but that just gets us into the authentication phase that gives us a token. Now how do we use that token, and does that token lives with us for the life of the application per user, per instance? What does it authenticate I guess...?

**Vittorio Bertocci:** Right. So let's dig deeper. The token that you get is typically per user. That's to say

that it is the user that goes into the identity provider. It somehow authenticates and it gets a token of what describes that specific user. For what goes in a lifetime of the token, that's typically decided both by the identity provider who is the ultimate arbiter of that, the token that typically includes wide expiration information.

**Carl Franklin:** Okay.

**Vittorio Bertocci:** So if the identity provider says, yes, I certify that this guy has three feet long hair but I can certify that only for half an hour.

**Carl Franklin:** It's saying he's getting a haircut.

**Vittorio Bertocci:** And after that the token will expire, and then also the application has to say that the default is the moment in which you get this token, you drop the cookie and you create a session. So from that moment on, the application doesn't need to keep sending you a token. You keep it in your belly and you have it. But you can certainly have control about the session that you created out of it. So for example you can decide that if the user is inactive for five minutes, you want the session to expire regardless of the fact that the token may last longer than five minutes and this is under your complete control. The reason is the pipeline of that in which these documents is first requested. When you hit the application, the application will realize you're not authenticated and will redirect you following the protocol. Then once you actually get the token, there is one stage where you get to this token, you understand which kind of token you are talking about, for example, X509, XML, and you call the appropriate handler. We are in a set of classes that security token can handle it. We have all the defaults for the most basic ones, but you can certainly write it on if you want to do something exotic, and there you will check the signature, you will verify if direction is similar, and then you will extract their claims. Claims that are, they are to say statements about the subject made by the identity provider. So those are the facts that you want to know about the user. The answers that your application needs to get for all users or customizing the experience and I'll say a bit more later. So you get to this list of claims. You go from one stage in which you look at those claims and you decide which ones you want to keep. So for example, your identity provider is a bank and in most of the claims there is a blood type claim, you certainly don't want to hold the bank authoritative about that so you will just ignore. We will stop this blood type claim here and you'll just allow every finance to go forward. And then there is one last stage in which you can actually perform authorization of that. Remember we're still outside of the application. We have yet to detach your custom code, and here Windows Identity

Foundation provides a hook. We give you a specific class, which is the Claims Authorization Manager, which you can subclass for injecting your own logic and you can go from the last basic of things like you can adjust and replicate the classic ASP.NET authorization logic, or you can even use their per diem, the same logic because we are compatible with the other model. So you see we are using ASP.NET IsInRole or the syntax authorization in the web.config, it will work also with claims or you can use this class for processing claims for things like what I've said before like if you're serving content for which you needed it to be all that are meant to have into one year, then here you can actually make the check. Of course, I'm talking about things like electrons and similar to watch where you're thinking.

**Richard Campbell:** So what sort of claims are we talking about here for folks who aren't familiar with claims?

**Vittorio Bertocci:** Claims that can literally be anything. In the context of IT, the most common claims will be the one that reflect, for example, your position in your organization. So you can have claims that describe the groups you are a member of. So you can be an administrator, or you can be in a chair and things like that. So typically in business-to-business, those are the most common kind of claims. But then they can also be pure attributes like for example you may have your name, your birthdays, your shipping address. So things that describe you rather than tell you what you belong to. And then you can have roles, explicit roles and you can go as deep as you want. So for example, in certain occasions you may even have permissions in the claims. Permissions are kind of a delicate point because permissions are typically something that are expressed by an entity that knows already about the results, and the identity provider can also be a barrier removed from the results. In other words, if I'm accessing a certain SharePoint on your organization in my identity provider here, that's not going to necessarily know about like the title of a document so you cannot expect my identity provider to provide direct permissions. It will provide the descriptions of what it knows about myself like my role and organization I belong to, and then it will be your system that will transform those claims into claims that are more descriptive and more useful of my access to the results because your organization knows about the results, and that's one of the beautiful, beautiful things of this architecture. But this is just service orientation applied to identity. I'm sure you remember the time we had service orientation. Here everything strives to give you autonomy. Everybody that knows about something, or about resource, about the user that's similar, has the chance of trying to gain this piece of knowledge without having to have strong dependencies on others. So

identity provider knows about users. It will talk about users. Resource providers know about resources and it will express logic about the resources and all this happens in complete independence. So the resource doesn't need to handle the credentials of the users like very often today it happens. Today you see very often that you have duplication, that you have synchronization problems that you have spread the credentials to the same user around them, then once this user gets the commission it can still access that around. So with a direction claims we solved that. But we also solve the issue of having to know from the source about other sources because typically there are sources that change even more often than users.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Think how often you hire or fire people, and how you often you change titles or location of documents. With all these claims system, you cannot have everybody owning its own pieces and everybody doing their own drop-ins with complete independence from others which is just great.

**Carl Franklin:** This portion of .NET Rocks! is brought to you by our good friends at Telerik who brings you the RadControl suite for Silverlight. Are you already playing with Silverlight 3.0? Then you might have started using .NET RIA Services, Rich Internet Application services, which makes data operations a whole lot easier especially for a line of business applications. So check it out. Our friends at Telerik are again ahead in the game tapping on the new benefits of Silverlight 3.0. The RadControl suite for Silverlight now fully supports .NET RIA services in domain data source. So if you're wondering what's in it for you, the answer is pretty straightforward. You get completely codeless binding to RIA services, impressive validation support on the client and on the server. Your customer will also be pleased to sort, filter, and page data much faster as all data operations are now server-side. Besides, the suite also offers out of browser support, and as you might already have heard, the first commercial 3D chart. Check out the Telerik Silverlight suite at [www.telerik.com/silverlight](http://www.telerik.com/silverlight). Don't forget to say thanks for supporting .NET Rocks!

**Richard Campbell:** Okay, before we go too far down the federation angle, I want to get there, but I want to pull back to some more code related things specifically what does it look like to a developer to check a claim. So if I've got a guy who's got a claim of being an HR and I have an application that's only suppose to be accessible; or say more a menu item that's only available to HR people, what do I have to go to enforce that?

**Vittorio Bertocci:** All right. You've made an excellent distinction. There's one point in which you just want to block the access of application altogether, like if you're not HR I will not allow you to see the entire page. That is achieved using the mechanism of a dimension before. You have one class, the Claims Authorization Manager, that typically looks at your web.config. In your web.config, you can say this source, which could be the page, can be access only by people that brings this claims type with this claim value. While in your case the claim type would be group and the claim value would be HR, it's that if you go all the way to that menu what happens is that from your cause, so finally we enter your application, you needed to retrieve the value of claims and that's really, really easy to do. So if today you are a .NET developer and you wanted to know about the identity of your authenticated user, what do you do? You just go on the thread, you get the current principle, and you obtain from that IPrinciple which is a built-in interface in the thread, from that you extract an IIdentity and there you will have information about the user with the IPrinciple you make IsInRole. So we have said that with Identity Foundation we build on the same mechanism. So on the IPrinciple, get the subclass by another interface, which we call IClaims Principle, which is still living in the same place so you can still retrieve it by doing traditional Current Principle from within your application, but you just have to cache it to two IClaims Principle and there you'll have all the things that you had before like if you wanted to call IsInRole, you can still call IsInRole, but if instead you want to use Claims you just extract the IClaims Identity Interface, which again if you would have a normal IPrinciple you would extract and IIdentity, here you have an IClaims Principle, you extract an IClaims identity. This is exactly what you would be normally writing. But in the result now, you have a collection of claims that are completely agnostic from how you have obtained them. It doesn't matter if you got them from a certificate or from some of the tokens. As a developer you don't see it. You just query this collection typically with a LINQ statement. So for example, if you are searching for the age claim, you just have age value equals, and then you could have something like from C in Claims Identity.Claims where C.Claims equal equal age, so let's see that value.

**Richard Campbell:** Nice.

**Vittorio Bertocci:** This is pure LINQ in its easiest form like in a typical select that you would do in T-SQL. At that point, you just have your value claim and you can decide the interface of that if you want to show that menu or not in a classic ASP.NET fashion.

**Richard Campbell:** So now I started out thinking, oh great, I can stay with IsInRole which I've known before even .NET, but I like your LINQ approach

better. That seems like the more modern way to go check a claim specifically or collection of claims.

**Vittorio Bertocci:** You make another very good point. IsInRole is familiar to a lot of people and of course, of course we had to maintain its support because in many scenarios all you need are roles because many systems today are role-based. However, with claims you can be more sophisticated. So you have a choice of moving forward. With IsInRole, you would not be able to check like if somebody is spending limit is higher or lower than a certain threshold. Instead, with LINQ, the thing that I've just mentioned, what I extracted is the value of the claim which could be anything. It could be a date, it could be an Int, it could even be a structure of data. At that point, if you have logic that is more sophisticated than that string matching, because in the end IsInRole is just very fine with a certain string that belongs to a collection. Certainly I can do stuff that is definitely more complicated, and actually complicated is the wrong term, I'd say sophisticated, that's to say it again. I can never have very complex business rules. Like for example, if the spending limit there of somebody is X and it's a certain time of a month, because we know at a certain time of the month we're going to pay salaries so our liquidity, our cash flow is in certain condition, and if the guy comes from a certain department than X, otherwise Z. For all these kinds of things typically you'd be force to do this in a way that is dependent from the kind of condition that you're getting, you'd be reaching out from various databases in your system so that if you take your application and you move it, for example, in the Cloud, then you would have no chance of reaching out for these databases and similar. Instead here we have everything that's nicely traveling inside one compact token and we have one single syntax that you can use regardless of the way in which you obtain this token. And so this empowers you to do more and with less dependencies, which again is the key of evolution. I hated it because even in those there's a chance that you are wrong of course, but I'd say that now that finally we have this tool in a couple of years I'm sure that you've seen much more like we move it to the next level that's to say that today IsInRole remains substandard.

**Richard Campbell:** Yeah.

**Vittorio Bertocci:** As soon as we offer this more sophisticated stuff, nothing will hold us back from actually having more standard things in which you can express more complex and more descriptive policies that are actually in front of your application.

**Richard Campbell:** And where this gets really interesting is in the actual federated side of things. The idea that I have a contractor who's working in a different organization doing HR work for me, I'm able

now to give him some kind of identifier so he would have the privileges of being in HR without actually having to be a part of my domain per se?

**Carl Franklin:** Yeah. This is where I start to lose it.

**Richard Campbell:** This is the hard part now, right.

**Vittorio Bertocci:** That's exactly correct. As long as your user can, with this token, obtain the correct identity provider, then your application will just process the token and it will be blissfully happy about it. Now, you cannot chain the process of obtaining a token crawling through federation relationships. So your contractor that will belong to some organization, with this identity provider, they'd be in a relationship with your identity provider so that when the contractor lands on the page of your application their application will redirect it to its own identity provider page, which in turn will redirect it to the contractor page. The contractor will get a token from his own organization containing a description of whatever is needed for bigger organization on the other side will bring the token to the other identity provider and once the system will run some transformation logic like, okay, I recognize you're coming from a partner and I will transform your senior inspector in my HR general list, and then this person will be able to land on your page and use your application using roles that make sense to your environment rather than that because your HR general list maybe associated to a number of privilege which of course the contractor's original environment knows nothing about, and while it is completely obvious how this hurts when you cross organizational boundaries, it's interesting to see that it can work also within your own organization. So imagine for example your HR department and the Finance department, and imagine all the classic permissions that you got in HR so an HR general list that can see all the personal details of the people that work under his own care. Now we mention that there is somebody in Finance that occasionally gathers to make the controller function. So you guys have to verify and investigate that somebody is actually making incorrect use of company resources, so expense knocks and stuff like that. So this person will have to dig on specific employees in some way in which HR would because HR is part of this investigation.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Now, with this system that you can actually have some authority, authority for HR and the authority for Finance that is separate with each other so that specific controller can be declared as controller by the Finance and you can also provide for example the list of names of employees that this guy is investigating, and when he learns of that side therefore HR can transform those information like this

guy is in the role of investigator, or these guys can transform those into privilege on HR department only for that people so that when he learns of the application in HR his permission will be exactly like if somebody is working in HR but it's just for these people.

**Richard Campbell:** Right.

**Vittorio Bertocci:** So you as an application developer for HR, you don't have to worry about the special case of, well, every time I get an HR employee but sometimes I get a Finance controller, you don't, you just go ahead with your business rules for HR systems and it's the infrastructure before that that takes care of giving the correct claims to the user so that your application doesn't change. You don't worry about the borderline conditions. You just execute on your business logic, and again this is huge because the moment in which you start worrying about all this stuff you achieve a level of autonomy that truly enables you to use all the possibilities that you have around. If you now have to take again this application and move it to another datacenter, move it to another department, move it to a host, move it to the Cloud or wherever you want, the logic comes with application. It's the fact that you didn't have to treat borderline cases but you are purely on point with your business function, it enables you to do that with much less worries than in the past because you know what happens when you take one up and you change something. Typically you get the call from a subsidiary on the other side of that world where somebody will say, hey, all my staff stopped working and you didn't even know that that application was having an influence on them.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Like this we can finally break all those ties and have everybody just worry about its own part.

**Richard Campbell:** But it strikes me that the challenge here is building the right claim that that finance guy shouldn't claim himself as an HR guy, but rather claim himself as privileged enough to look into people's personal data.

**Vittorio Bertocci:** Exactly. In fact, once you get the two application level then the claims that you will have will be describing your access level or any way this specific roles format application. So this will not be in an organizational point and that's exactly the point, that the application doesn't intend to worry about that level. It has to worry just about enabling the specific operations that we are giving. In other words, let's say that you are the developer that is writing their web service that is accessing the payroll

of people, or the history of their payroll which is more on the HR function.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Then you needed to worry about how to perfect those specific operations. So typically you'll expect something like roles or privilege. This system will enable the finance guy to obtain those specific requirements without you, web service developer, worrying about trying to understand those. Like imagine the warning which you just get the username and password in your web service. From that username and password, then you would have to make some internal investigation like finding out where this guy belongs to, finding out the fact that he is now acting as an investigator, finding which employees he is allowed to call and so on and so forth. Instead, with the system in which you cannot trust far more claims along the way, you can put all this intelligence where it belongs to because it's an agreement between HR and the Finance that any investigator should have certain powers of accessing the backend. It's not the master of this specific web service. It's something that is decided at the department level, and then from the department to the specific application it's a matter of the application and the HR department to decide which roles have which privilege. Again, you are pulling from the application or the logic that you would normally have to write if you just have the username and password and you're putting it to the right places. So in a point of contact between the HR and the Finance, you put the intelligence by saying yes, I have the notion of an investigator's role and I know what it means for my area. And then you get at the application level the notion of yes, I know the roles of users of this application and I know what are their privileges associated to that. So that you just worry about that as a web service developer, and instead the administrator of the identity provider that believes in HR worries about managing the relationship with Finance and all. So notice the different competencies that are here. As a web service developer, you should know about let's say company politics or similar. You just do your job of locking your web service and handling the permissions. The fact that the investigator should have a certain power is something that you leave to higher-ups like it's a negotiation between the VP of HR and all of the people below him, and the VP of Finance and all the people below him of course depending on the size of the company like probably here the VP doesn't care about the business logic. But you see my point that with this you can actually bring staff where it belongs to. That's to say that we're at a transitional level or at the application level.

**Richard Campbell:** Well, this is a long way away from checking IsInRole and "HR."

**Vittorio Bertocci:** Right. Well, you can still do that in a certain – like for example, there will be situations in which you still want to do it. That's to say that in the end all this chain may just prepare you to be able to just do that at your application. Your application is your logic ease. You have to write `IsInRole`, that's fine. As long as all the people that lives around you did their job so that the people that should access your application actually gets the best plane.

**Richard Campbell:** I'm still trying to envision the UI where the VP of Finance calls up the VP of HR and says, hey, I need to get my guys investigating these people, and I'm trying to figure out what the interface would look like to actually go in and say for this finance user, give him the best, get in privileges on the following employees. Is that something that's part of the ADFS, the Active Directory side of things? Or is that actually in the application?

**Vittorio Bertocci:** Right. Actually I shouldn't have used the term 'VP' because I don't think that they would get to do any UI. They would probably get their technical decision makers and tell them make use of that. The UI, as you correctly points out, would be largely in ADFS 2.0. In ADFS 2.0, you can establish a federation relationship that you can say, okay; I want to be able to accept tokens from this separate entity provider. So if you are in HR you can say yes, I want to accept tokens from the Finance ADFS, and conversely you can also say the opposite like yes, I'm willing to give tokens when I'm ask there to give it to that specific endpoint. So Finance can consent to give the tokens to users that are trying to access HR department. These are just clicks like remember when we describe the developer experience. That main experience is very similar. You just point your ADFS to make a data of the other ADFS. You make a couple of clicks and the relationship then is done for you. Like other details that you are suppose to write in ADFS 1.0, now they are all automatically done for you. So the certificates and endpoints, everything is automatic. Then once that is established, you decide which claims you want to send on one side like the finance guy can say you can. When somebody is asking for a token for going to HR, the things that I want to say about you or your group, so if you are an investigator it will show up in your group, I want to say your name, I want to say all the things that you want to say about the user in that context. Instead, when you go on the HR side, what you get there is transformation rules that says that you know the set of claims that you're going to receive from the other one, and you can say what you want to do with those claims. So for example you can have rules that when they receive a group that says in this group this guy is an investigator, then you can actually trigger all the rules in that different claims in the token that you will

create so that you can actually make this mapping happen.

**Richard Campbell:** This is still a complicated piece of coding here. I want to be very careful when I build this to make sure that it's entirely data driven because both the users and the claims live outside of my application and they're going to change so I better build in my app the ability to go in and pick up the latest set of claims and then ask the administrator of the application how do these claims apply to this app.

**Vittorio Bertocci:** So here for what I've described, there is a no call involved. This is a purely administrative function. You are getting claims from -- like the way in which your transformers claims is through a claim inference language which is a scripting language and that's completely independent from the specific app like this is between one department and the other department.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Then in the application, you decide what to do with those claims.

**Richard Campbell:** Yes.

**Vittorio Bertocci:** But you get the latest claims every time because when somebody comes with an application it needs to authenticate it will obtain a token and this token will always be fresh.

**Richard Campbell:** Right.

**Vittorio Bertocci:** That's to say unless you're in the context of one session, this token will be fresh. At the same time, remember when I describe the experience of developing the application. When you create the reference, the STS, you get the release of claims that you will get from this specific identity provider so that then you can actually use them directly, and this operation also changes the configuration of your app and provides a list of claims so that you're going to get, or that you require in your application, so that you can have a number of automations. Like for example, one very simple example that shows how you can use controls for influencing the behavior of your UI depending on the value of those claims, the list of claims that you're getting is actually coming from a metadata of the identity provider. So you can actually have tooling that helps you in this respect, and there are certain changes. There is in another entry in the menus that are describe in Visual Studio, which is a refreshment of data in which you can actually reach out and redo the configuration so that you essentially change the release of claims and you will actually reflect those changes in your app.

**Richard Campbell:** Right.

**Vittorio Bertocci:** So again, the tooling really helps you in this respect.

**Richard Campbell:** And I'm trying hard to think through being able to deal with claims changing at the ADFS level and not having to recompile my app to deal with that. So in my application, I'm able to go off and make a metadata request to get a claims set when I start up just to say am I going to be able to operate, like is the identifier for HR still out there so I know when to run the HR features?

**Vittorio Bertocci:** You could do that but that would not be something that you would do in the runtime. You would do it at the design time. And here is actually a...

**Richard Campbell:** But Vittorio, the important part there then is if we're making major changes to claims, we are talking about recompiling the app.

**Vittorio Bertocci:** It depends by where you're putting your logic because if your logic is just for access, you are putting stuff in your Claims Authentication Manager, then you don't need to recompile the app. You just change -- there's something in the web.config.

**Richard Campbell:** This is always about granular roles. To me it's all about do I pop the menu item or not, and I can see it's not thinking about being able to have finance do investigations on people and so I've got to go and retrofit those claims in and then have the application deal with it and probably make a couple of iterations on that. So I'm just worried that my app is not going to function correctly should securities get in there and mess with a bunch of claims.

**Vittorio Bertocci:** So I wouldn't worry about that specific aspect because then you you're to have exactly the same worry when you are developing applications today because if you are relying on anything like the Active Directory schema, then that would be exactly the same. That's to say that you would not expect the claims to change often especially if you are taking stocks from your own identity provider.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Like if you're an HR app and you're getting claims from the HR STS, then if there is a major change, if one of the things that you needs to be managed exactly like if you were to do it for the, exactly as you would do it in an Active Directory case, so claims are not the claims types, are not something that you should expect to change very often, and

when they change you should expect that you know it upfront because what you said is a pure change management like for everything else.

**Richard Campbell:** Right.

**Vittorio Bertocci:** But there is something that improves because you as a petition developer is decoupled from the changes that happen on things that you don't control like, for example, if for some reason you make a management acquisition and you buy a company that does only finance, it then becomes your own finance department. So this new company has a whole lot of different claims that more or less describes the same things but we are different.

**Richard Campbell:** Right.

**Vittorio Bertocci:** Now, you can manage this change at the identify provider to identify provider level.

**Richard Campbell:** Right. So I should be able to map the claims of the acquired company to the claims of the existing company.

**Vittorio Bertocci:** Exactly and that happens at the administration level. You as a developer are protected from that, because you as an HR application developer, you are still dealing with the claims that work in the HR department.

**Richard Campbell:** Right.

**Vittorio Bertocci:** So if a mapping needs to happen, it will happen even before it touches your application.

**Richard Campbell:** Yes.

**Vittorio Bertocci:** So you have just mentioned about change management, you are definitely not worse than everything else in the company and if anything, you are better because there are certain things that you should not worry about and finally you can afford not to worry about like for example this kind of infrastructural things.

**Richard Campbell:** Yeah, there are bigger problems to worry about, but it's interesting in my mind to think probably the first time ever, if I'm getting into claims-based security, I've got to talk to my security guys as a developer and say, "Look guys, I don't want you to ever remove an existing claim. I want you to add to them. We may add more granular or we may change configuration, but if you pull claims out you break me," and that is to me a very new idea.



**Vittorio Bertocci:** Well, again, I see how you can maybe perceived it that way, but in the end this is all exactly the same.

**Richard Campbell:** Yes.

**Vittorio Bertocci:** Because the claims are just a way of obtaining answers to all the questions that your application has about the user. Even before claims, you are obtaining those answers somehow.

**Richard Campbell:** Yes.

**Vittorio Bertocci:** We are all querying a profile. All are querying Active Directory.

**Richard Campbell:** Yeah.

**Vittorio Bertocci:** All are querying different database. All are managing various systems like maybe the user was coming with an X509 and the subject of the certificate was use it as a key for querying some database and so on. All falls to data sources like resources of truth that we call them in identity. It could have change at any moment. We have no warning exactly like the one you're just saying with minute.

**Richard Campbell:** Yeah.

**Vittorio Bertocci:** More and worse because typically when you don't establish a single way of obtaining information, but you are aggregating information from different forces, some of the sources maybe that volatile like for example the profile that somebody put together for some reason that you are using for another reason, then once the parallelism goes down then all your systems will go down as well and nobody knows why because nobody realized that somebody was making let's say an abuse of another system. Instead here, you have a clear tracing from where stuff comes from. You have one specific acquisition point that lives outside of your application in which you get all the information that you need. So again, I would argue that now its much easier because if you are to manage the change, you have one single place to go and you have very clearly a way of troubleshooting what is going on. While in the current situation you are in deep, deep trouble because those changes are going to occur in many different places. You may not be even aware of where those places are or who the owners of those things are. So again, claims, so many of you now are bringing some order in a place where there is no order. In fact, if I were in your shoes, I wouldn't have to be worried about those specific stuffs. I would be relieved that finally there is somebody on point for many those changes. After sometime, since there is somebody on point, the probability of those changes that will happen without warning is much lower

because now it's clearer what you're going to do with that information. Let's say that somebody creates a page with phone numbers of employees just for one specific conference like everybody is going to have conference and it's useful for the conference organization to have one page with those phone numbers. And then somebody else starts using that page and making screen scraping for getting the numbers. I'm not making this up. I've seen this happening.

**Richard Campbell:** Sure, yeah.

**Vittorio Bertocci:** And this page gets off and lives in the internet for a couple of years until somebody decides, you know what? We have a way to many pages. We need to consolidate. They start to go through all these and they find this page and they say, hey, look at the crowd. This is about a conference of three years ago. So wait, enter the commission gap. And then a number of other apps that were lying, they all start to go down. With that here, it won't happen. The point here is simply that when people think about authentication, they usually think just about the credentials like, yeah, he's an impostor, and now he has all these claims. Oh, maybe it's something that I have to manage. That really, it's something that you already had to manage before. It's that finally now you can have everything in one single, nice package and you know where to place the blame when something goes wrong.

**Richard Campbell:** Well, I think the challenge is building good code that fails well when there are security problems. Too many times I've seen apps where because you couldn't connect to the domain, the error message that comes up is still like invalid password or you failed to authenticate implying that user did something wrong when actually the infrastructure is broken, and that to me is -- a well written app would actually say, hey, you know what? I'm relying on a claim that I just went and check my services and that claim doesn't exist anymore so there's no way, there's nothing you as a user could ever do to be successful here.

**Vittorio Bertocci:** I completely agree with you. There's always a finer balance between being descriptive and giving too much information that I gave in the last situation...

**Richard Campbell:** Yes.

**Vittorio Bertocci:** That somebody is trying to break into your account. If you tell them that their password is wrong but they didn't say anything about the username, you maybe giving them an indication that the username is correct. But apart from these little details, I absolutely agree that the error messages of today are very unhelpful and I believe

that claims can help with that; that is to say is the analogy of a richer information. You can actually be more descriptive about it and guide user to see if there is something that they can do, or as you say like get another piece, you really have specific powers. Again, one interesting thing in claims is that since everybody declares upfront what are the things that are needed before operating, you may just prevent this situation altogether.

**Richard Campbell:** Right.

**Vittorio Bertocci:** In other words, if your website is asking for a certain list of names from a certain identity provider, you follow the system, follow the protocol for you and reach for better identity provider asking for a certain set of claims. That identity provider will -- if it cannot give you the new claims -- it can actually give you a meaningful answer about that.

**Richard Campbell:** Yeah.

**Vittorio Bertocci:** Like you mention this, if the identity provider would give you the document anyway ignoring the fact that you're asking for claims that it's not giving you, then the application would be hit by that and would adjust -- the replication would be good enough to know why you don't have this claim. The best it could do would be to tell you, you know, I need this claim and you did not bring it to me and you're talking so I can't go further. But instead if you're going to the identity provider asking for a reclaim, the identity provider is then saying that hardly all the claim would default so now the identity provider knows why it doesn't tell the order claim any longer. So it can actually give instructions like for example say that there was a breach. You know, sometimes it happens with, for example, credit card numbers are stolen. Unless, let's say, that you are actually calling a system that needs a credit card, you go back here to the credit card company that tells you that it's on the identity provider, you try to obtain a certain security number associated with your credit card and the credit card system can actually give you a message saying, hey, we're really sorry but we can't give you this specific claim because your card number is among the ones that may have been compromised. So you have to go call this green number and we can renew the card for you. Instead, we would be sealing the other system in which you hit an application and the application somehow tries to find that out on its own, then it may not have enough information to give you a proper explanation of what is going on and what you can do. So again, it's more and more about control. Controls are the users, controls are the applications, controls are the identity providers.

**Richard Campbell:** Absolutely. Hey, Carl, you still there?

**Carl Franklin:** Huh, huh, what, what? Who's that? What? Who's -- what did you say? What? I'm sorry, I must have nodded off.

**Richard Campbell:** Oh, thanks for that. I thought we were having a really good discussion on getting to the....

**Carl Franklin:** You guys were having an awesome discussion.

**Richard Campbell:** Yes.

**Carl Franklin:** However, it's completely irrelevant for the kind of stuff that I do so I thought I'd just let you talk. I think anybody who's really interested in this is probably *Jonesing* for it. So where can we download it? Where can we get it? Is it just a part of .NET 4.0?

**Vittorio Bertocci:** The Windows Identity Foundation is actually based on .NET 3.5 SP2. So it can be used also with systems that are created with .NET 4.0, and in this moment you can find it under the name Geneva Framework. So you just go on your favorite search engine which is Bing of course, and you just write Geneva Framework and the first results will be the installation of the SDK. Also, if you want to learn more about this, we have a number of examples and one training kit for developers that provides you with certain hands-on lab that you can walk through for learning the ropes and also for doing stuff with that is more advanced like Delegation and similar functionalities. And again, in any case you just search for Identity Developer Training Kit and the first result is pretty much always it.

**Carl Franklin:** Awesome. Vittorio Bertocci, thank you very much for joining us for this hour plus.

**Vittorio Bertocci:** Thank you, guys. It's a pleasure. It's really interesting for me.

**Carl Franklin:** And we'll see you next time on .NET Rocks!

[Music]

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