



Hanselminutes

Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

Text transcript of show #85

October 16, 2007

EarthClassMail.com - Moving from LAMP to .NET 3.5

Scott chats with Matt Davis, architect at EarthClassMail.com, about their move from a LAMP stack (Linux/Apache/MySQL/PHP) to .NET 3.5. What's working, what's not, and what kinds of issues are they running into as their architect their solution.

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



Our Sponsors

 **telerik**
deliver more than expected
<http://www.telerik.com>

 **nsoftware**
<http://www.nsoftware.com>

NET 
DEVELOPER'S JOURNAL
<http://dotnet.sys-con.com>





Lawrence Ryan: From hanselminutes.com, it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman, hosted by Carl Franklin. This is Lawrence Ryan, announcing show #85, recorded live Monday, October 15, 2007. Support for Hanselminutes is provided by Telerik RadControls, the most comprehensive suite of components for Windows Forms and ASP.NET web applications, online at www.telerik.com, and by .NET Developers Journal, the worlds leading .NET developer magazine, online at www.sys-con.com. In this episode, Scott talks with Matt Davis, architect at EarthClassMail.com.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes and today I'm sitting here with Matt Davis, the architect at EarthClassMail.com. EarthClassMail was talked about a little bit here on Hanselminutes when we talked to Tim Ferris, the author of "Four-Hour Workweek," and he uses it to dramatically lower the amount of information that he has to absorb, the amount of mail he has to touch. So, Matt, what is EarthClassMail?

Matt Davis: Basically, it works kind of like mailboxes, etc., for the 21st century. When you sign up with us, we'll give you a virtual P.O. box and you change your address to us. We have a number of different processing facilities around the country and different addresses at which you can receive mail. You change your address to one of our addresses and then any mail that we receive, we scan the envelopes and make them available to you online. So, you'd log in just like you would to your Yahoo mail or your Gmail account and you see all the external envelope images of any mail that you've received. At that point, you can do one of a few things. You can either ask us to open it and scan it and in our secure processing facility we'll open that up and scan the contents and make them available to you as a PDF or you can say, "This is junk mail. I don't want it. Just shred it and recycle it," whichever you choose. The other thing you can do is have it shipped to you. You can have it shipped to you without us opening it or we can open and scan it and ship you the contents, either way. It makes your mail available to you basically anywhere you are in the world.

Scott Hanselman: Let's say I get 10 snail mails that day. I can go and say, "Shred, shred, shred, and these 7, mail them to me." You'll batch them up into a big envelope and mail me the whole envelope?

Matt Davis: That's correct.

Scott Hanselman: Do you scan it front and back so we see the whole thing?

Matt Davis: Yup.

Scott Hanselman: Wow. So, you could see whatever notes are written on the envelope or whatever.

Matt Davis: Yup, everything that's in the envelope. They basically have no exceptions policy on there, so I mean if you get a piece of junk mail, if you get your Publishers Clearing House thing and you want to see every little scrap of paper that's in there, we're going to scan it all.

Scott Hanselman: Really? Even like flyers and kind of like Domino's Pizza advertisements and coupons?

Matt Davis: Yup. It just makes it simpler for you to say. We just scan it all.

Scott Hanselman: I would suspect that you probably shred a heck of a lot of mail.

Matt Davis: We do. There is a monster shredding pile over there at the warehouse.

Scott Hanselman: Really? I have this image in my mind of just like robots, Sony robots doing all of this work. Is it that or is it people?

Matt Davis: Right now it's people. We've just recently gotten a nice funding block that we're just starting to dip our toe into the waters of automation, so we've got some really interesting equipment on order that's going to help us get started with that. If you look at the website, you'll see things about mega sorters and we've got some prototype robotics designs and things for doing like really mass quantity storage of mail.

Scott Hanselman: Wow! At some point, it will probably be faster for the post office just to hand you the mail than to deliver it through the post office process.

Matt Davis: You know, that would be a wonderful day.



Scott Hanselman: That would be cool. So, this sounds like a phenomenally huge amount of data along with little tiny bits of data in the sense of I'm just envisioning there's me and I'm some GUID or some identity in a database somewhere and then there's unique numbers that identify all of my different pieces of mail and these are all very small amounts of information and then these huge PDFs and images of my mail.

Matt Davis: Yup.

Scott Hanselman: So, all these little bits of information that point to all these huge amounts of information. Are you using SQL Server or are you using MySQL? What's your guys' technology?

Matt Davis: The system as it exists as of this recording is running on a LAMP stack.

Scott Hanselman: Tell our listeners what LAMP is. What kind of Windows is that?

Matt Davis: Yeah, okay. Linux, Apache, MySQL, and PHP.

Scott Hanselman: Okay.

Matt Davis: The system as it stands today was built by a Russian contracting firm. I don't want to make any comments about quality, but let's say it, you know...

Scott Hanselman: It was built...

Matt Davis: The original system was built quickly.

Scott Hanselman: Okay.

Matt Davis: Yeah.

Scott Hanselman: So, it's a quickly built LAMP stack.

Matt Davis: Yeah.

Scott Hanselman: Based on PHP.

Matt Davis: Yeah. It's taken quite a while to get that stable. It's quite stable as it is, but it doesn't give us a great platform for growth into the future. We are just putting the finishing touches on our .NET port. Actually, it's not really even a port. It's

a rewrite. It's a complete rewrite with not really a lot of thought to trying to preserve any of the architecture that was there. Basically, the UI looks about the same and that's it.

Scott Hanselman: Was the original LAMP stack pretty straightforward? I mean web browser hits Apache web server and Apache web server writes directly to probably the file system and/or MySQL database?

Matt Davis: Yup, that's correct. As you can imagine, that presents a number of scalability problems.

Scott Hanselman: Let's talk about that architecture. Are the issues primarily just that it's a straight three-tier architecture? It looks and sounds more like a two-tier architecture.

Matt Davis: Yeah, it really is. That's one of the biggest problems.

Scott Hanselman: So, you don't have an intermediate anywhere?

Matt Davis: The funny thing is that they did everything in PHP. There are little backend daemon processes and things. There are actually cron jobs that are running PHP scripts on the backend. They did everything in PHP.

Scott Hanselman: Well, you know, when all you have is a hammer, everything looks like a nail.

Matt Davis: That's right.

Scott Hanselman: So, is MySQL holding up under the strain or is that an issue?

Matt Davis: Actually, we haven't had any trouble at all with the MySQL stuff. The real problems that we're having are just memory and mostly implementation details. I have no doubt that we *could* have rebuilt the system on LAMP. I don't want to get a bunch of flaming emails from LAMP lovers out there. There's no doubt that we could have rebuilt the system on LAMP and been totally fine, but the .NET platform gave us a lot of really interesting new bits that we didn't have to build.

Scott Hanselman: Was that a hard sell? I mean I assume that there's some boss somewhere above you who has to make those kinds of decisions.



Matt Davis: Honestly, I don't think it was a hard sell. I actually wasn't even involved in that decision. I was hired on after they decided they wanted to port the new system to .NET. They were looking for a .NET architect that was used to building fairly large systems, so I applied and got the job.

Scott Hanselman: So, you're doing this on .NET 2.0 and Visual Studio 2005 or...?

Matt Davis: Actually, we decided to go ahead and bite the bullet. We're building on Orcas.

Scott Hanselman: Really? So, you're building on Visual Studio 2008?

Matt Davis: Uh-hm.

Scott Hanselman: Now, does that mean just the IDE or are you using .NET 3.0 and 3.5?

Matt Davis: We're using .NET 3.0 and 3.5. I think the number of standard SQL statements that exist in this piece of software probably could be counted on one hand, the entire thing is powered by LINQ to SQL.

Scott Hanselman: Really?

Matt Davis: All the data tier is powered by LINQ to SQL. We're using WCF to communicate between the tiers. All the operation software is written in WinForms and WPF.

Scott Hanselman: Really, WPF? Are you doing ClickOnce?

Matt Davis: Unfortunately, we have some real problems with ClickOnce. We have to do a little bit of native code Interop to talk to scanners and things like that.

Scott Hanselman: Oh, right, right.

Matt Davis: ClickOnce doesn't play -- you can make it work, but the way our... The other problem with ClickOnce that we had is you can't deploy to a machine, you deploy to a user.

Scott Hanselman: Oh, right.

Matt Davis: Our operator workstations are kiosks so we may have a number of different

operators sitting at a given station at any given day and so each one of them has their own Active Directory login, which means they have multiple copies of the software given that we have some queuing software and stuff that's running in the background as well as a Windows service. It didn't play too well with ClickOnce as it stands today unfortunately.

Scott Hanselman: Wow. So, there's a lot going on here. I'm trying to think what we should talk about first. So, let's start at the backend and work our way up to the glass, up to the monitor.

Matt Davis: Okay.

Scott Hanselman: LINQ to SQL, so you really kind of jumped in, both feet. Was that a hard sell? Sometimes there are shops that are very database-focused and they really think that SPROC's are where it's at. The logic is in the SPROC's. LINQ to SQL is really a powerful technology, but I wouldn't say it's SPROC-friendly.

Matt Davis: Yeah. Actually, I might almost disagree that it's not SPROC-friendly. You can make it work with SPROC's, but obviously the real power and sexiness of LINQ to SQL is being able to do everything right there in the code. I thought it might be a hard sell initially. My boss, the VP of Engineering here, he's an old school database head and so I was figuring that I was going to get some pushback on that, but really he just said, "Do whatever you think is right."

Scott Hanselman: That's cool.

Matt Davis: Initially, I was a little nervous about it and I had all the LINQ to SQL stuff locked away in a class. I didn't let anything leak out of that class so if LINQ to SQL just completely fell down on me, then I could go back and fix it, but it's been working fantastic. I mean we found a couple of bugs here and there and we had some pain going from beta 1.0 to beta 2.0. They changed the null semantics on us and a few other things, but in general it's worked out fantastic.

Scott Hanselman: Okay. Let's talk about it from a design perspective. So, you've got a bunch of LINQ to SQL statements. You said that they are working okay with SPROC's, so there are some SPROC's that you call?



Matt Davis: Actually, no. We've LINQed in a couple of SQL Server functions so that we can use them directly from the LINQ to SQL stuff, but no. Actually, we have zero stored procedures. We have like two triggers. I mean this is pretty much a code-centric application, which makes the deployment update play nice.

Scott Hanselman: You were saying just a second ago, and I need to be educated about this, LINQ to SQL is friendly to SPROCs, but you said also you had some SQL Server functions that you've LINQed in. How do you do that?

Matt Davis: Right now, it's not a very good story. You have to actually hack the DBML files yourself. The OR designer doesn't allow you to drag in system functions or at least I haven't figured out how to get it to work. There might be some trick, but I'm not a big user of the Server Explorer so I don't know all the different modes that are there.

Scott Hanselman: Sure.

Matt Davis: As far as LINQ to SQL is concerned, our system function or our system SPROC looks the same as one of yours. It's just a matter of how you address it in the DBML. Normally, if you have user-defined functions and user-defined SPROCs, you just drag those in out of Server Explorer under the OR designer surface and they just magically appear as methods on your data context and you can call them from anywhere you're using that LINQ to SQL data content.

Scott Hanselman: Oh. I'm going to have to educate myself about that. Okay. So, you said you had this stuff locked away in a class. You've got a bunch of stuff on the class and it's using LINQ to talk to the database. That's kind of the right hand. What's going out the left hand? What kind of objects are you returning out of there?

Matt Davis: This is one of the other things that was really sexy about LINQ to SQL. The interop story between LINQ to SQL -- we're using WCF, the left hand is WCF.

Scott Hanselman: Right, what used to be called *Indigo*.

Matt Davis: Yeah. I loved Indigo.

Scott Hanselman: It's a good name.

Matt Davis: Yeah. I still die inside a little bit when I call it WCF.

Scott Hanselman: Yeah, it's kind of sad.

Matt Davis: I was glad they left Silverlight as Silverlight.

Scott Hanselman: At least LINQ isn't called WDF or something.

Matt Davis: Yeah. LINQ is actually kind of a sexy name.

Scott Hanselman: It's the Q.

Matt Davis: It is the Q, you are right. It's all about the Q. So, anyway, yes, the left hand side is WCF. The interop story between LINQ to SQL and WCF is a little scary. I don't like to let my message contracts and my data contracts look too much like my database. I like them to look like more logical.

Scott Hanselman: So, you like to separate your data transfer objects and your business objects?

Matt Davis: Yes.

Scott Hanselman: But with a DBML file, you point your database at it and then kind of public class foo shows up if you've got a foo table unless you decide to hack that up. You can kind of get to where I was going. Do you let those objects out?

Matt Davis: Yeah.

Scott Hanselman: Or do you make other ones that look like them?

Matt Davis: I screwed around for about two weeks trying to figure out a decent way to let those objects out or to decorate those objects in such a way that I could control what they look like.

Scott Hanselman: Control what they look like on the wire or from the CLR perspective?

Matt Davis: Well, both, but more what they look like "*on the wire*."

Scott Hanselman: Okay, so you're talking about putting data contract and basically WCF attributes decorated on the generated...



Matt Davis: Correct, or trying to do it in a partial class on the side or trying to do it on a base class. I mean I messed around it for quite a while trying to come up with a combination that worked. Actually, they changed the story between beta 1.0 and beta 2.0. Beta 1.0 in the OR designer, you could actually apply any attributes you wanted to any of that stuff. They backed off on that and now basically there's no custom attribute support in the OR designer that basically a bit you can flip on any class that says serializable or not. If you mark it serializable, it adds the data contract and data member attributes to everything. It's all or nothing.

Scott Hanselman: Oh. So, the serializable bit that you're talking about is almost like "make this WCF friendly or not," but not the granularity that you wanted.

Matt Davis: Right. So, what we ended up doing actually is building just really simple copy methods that will do a deep copy of a given LINQ type into a given WCF contract type. So, we actually have completely separate WCF contract types to find. The really cool thing about LINQ to SQL was that you can actually plug that in, in your select statement. So, in the select call on the LINQ query, we actually call that copy method, so instead of like getting back a bunch of objects and then "FOR/EACHing" over them and making copies of the WCF objects, it actually comes straight out...

Scott Hanselman: Right, doing a bit form of marshalling.

Matt Davis: Yeah. It actually comes right out. We squirt the WCF bits right out the back of LINQ.

Scott Hanselman: So, then what's the point of the LINQ types at all?

Matt Davis: The LINQ types still have to abstract. I mean ultimately at some level you have to have something that looks like what's in the DB.

Scott Hanselman: I see.

Matt Davis: I know with Entity Framework and some of the other things that are coming out, the story is a little different. Entity Framework is interesting to me. I wonder if it wouldn't solve some of the problems, but I just don't have the time to look

into it since it's not shipping and the timeframe that works for us. I'll definitely be looking into it when it comes back. Anyway, LINQ to SQL, you ultimately have to have a class somewhere that looks like what's in the DB so that that OR mapper can do its job. The easiest way to get that is to use the OR designer, which is just the GUI drag and drop tables out of SQL Server.

Scott Hanselman: This deep copy, is this a reflection thing?

Matt Davis: No, it's just straight, stupid this field maps to this field copy, SQL, SQL.

Scott Hanselman: Is it manually written?

Matt Davis: Yeah.

Scott Hanselman: I don't know. I'm just brainstorming here. I'm trying to challenge my own assumptions as well as yours. You wanted so much control over the WCF types that you did that left hand/right hand, kind of this field/thatfield work, as well as having a whole separate kind of identity. What was wrong with just the "push a button and then it's available to WCF" serialized property?

Matt Davis: Well, some of the growth changes that we know we're going to be going through, I think trying to keep them isolated a little bit. Basically, what we did is we designed the WCF types to be a little more forward looking in a way to have a few features in there that aren't turned on yet. We have things that aren't even in the DB yet or we have things that we sacrificed some normalization to get something where if we tried to expose that directly through the contracts types, you'd end up with something that's kind of unwieldy. It actually works really well when you're talking to it through LINQ because you have some fairly decent control over back references and things that make them a little more queryable. It's just fantastically easy to do joins and things like that. I mean I look at the SQL as something that I can write in a fairly short line of a LINQ to SQL query. I would be 30 or 40 lines of nasty SQL to put it together and it generates it for me. You can watch the queries that come out every once in a while.

Scott Hanselman: Right, with the data visualizer.

Matt Davis: Uh-hm. Well, even without the visualizer, you can sort of see. I know, Scott, you've



put out that LINQ to SQL Query Visualizer and that thing is just fantastic.

Scott Hanselman: Yeah. Have you used LINQPad?

Matt Davis: I haven't. I've seen it.

Scott Hanselman: You got to check that out. It's basically a query analyzer. It lets you write LINQ and then see what comes out. It's a query analyzer-like experience where you type LINQ F5, type LINQ F5. You basically get your queries right inside of LINQPad and then copy-paste them over into Visual Studio.

Matt Davis: I'm assuming that's targeting LINQ to objects, not LINQ to SQL.

Scott Hanselman: No, no, LINQ to SQL, LINQ to objects, LINQ to whatever. It's a totally generic thing.

Matt Davis: Interesting.

Scott Hanselman: Yeah.

Matt Davis: I'd be interested to see how they're doing the class mapping stuff for LINQ.

Scott Hanselman: It would be interesting to see if your stuff works, the stuff that you're doing, where the objects actually pop out the side.

Matt Davis: Yeah.

Scott Hanselman: What about validation? One of the things that I thought was cool about LINQ is the idea that I could have like public partial class product and that's inside of the designer, that's generated, and the way that they get around not being able to edit that generated code as well as avoiding kind of uncomfortable object hierarchies for things like validation, like they could have called that product base and then you derive and have it on and validate it and things like that on insert and stuff like that, you're going to have partial method on validate and that's the method that gets called when it's time to validate a particular class. Have you done that kind of stuff at all?

Matt Davis: We've played with the partial method stuff a little bit. Unfortunately, we did most of our design on the beta 1.0 stuff where the partial method bits weren't in there yet. Since switching to beta 2.0, we've plugged in a few things there, but a lot

of that stuff was baked after we had baked our code. The partial method stuff is really interesting though because it is basically just like lightweight eventing.

Scott Hanselman: It really is, yeah.

Matt Davis: I mean it's really cool that the compiler doesn't generate anything. You only pay the price for that eventing overhead if you actually do something with the event. You implement a handler for it.

Scott Hanselman: Yeah, it's taking me a while to get my brain around it. If you think about how I've got this class and I say "data context submit all changes and if that class isn't cool, I want to know about it," how could that have been done? It could have been done with a pubsub. It could have been formally eventing, .NET events. It could have been derivation with a virtual method that gets called if it's overwritten or with partial methods where if the method is implemented, it will get called. How do you keep in track of validation? How do you prevent a foo object from getting into the database if it's not really correct?

Matt Davis: We have several layers of validation going on. Obviously, the first step is happening up in the UI and in the frontend of the service tier just doing standard, old school validations, ASP.NET validation things on the frontend and just normal checks on the backend.

Scott Hanselman: Input validation controls.

Matt Davis: Yeah, yeah, just basic stuff. Further back, we've got a lot of checks in the DB itself that are doing some consistency checking. There are few triggers that run looking for goofy things that we couldn't catch in easier ways, but for the most part basically we do the checking as the objects are being created because usually since there is that left to right where we're converting from a WCF type to a LINQ type, sometimes we'll do the checking in there too. One of the problems that we have with the validation, obviously when you split things across tiers that way, you usually have to do the majority of the validation upfront. We don't want to have to go all the way back to the DB tier to find out that something's wrong and then have to marshal some kind of an exception that's understandable to ASP.NET all the way back up.

Scott Hanselman: Right.



Matt Davis: The stuff that is going on in the service tier is mainly like the kind of the backstop for things that either didn't get through or were maliciously stuffed in through an AJAX endpoint or something that bypasses.

Scott Hanselman: So, ASP.NET makes a service proxy I assume, makes a WCF service proxy?

Matt Davis: Yup.

Scott Hanselman: It makes a foo object that is a WCF object and then passes it in to WCF. What do those methods look like and how do they map all the way back?

Matt Davis: Well, that's one of the places where we've kind of gone away from the Microsoft guidance a little bit and we're taking advantage of some of the flexibility of WCF. I'm definitely more of a "contract first" kind of developer. I like the idea of defining my wire format to an extent. I'm not quite as into it as Tim Ewald would be...

Scott Hanselman: Sure, but you like your control. You want your angle brackets a certain way.

Matt Davis: I do, I do. In almost all of the literature that I've seen describing WCF, everybody kind of downplays the message contract. The message contract is the thing that basically maps to like a WSDL operation.

Scott Hanselman: Which is funny because it seemed to me like a couple of years ago it was all about contract first.

Matt Davis: Yes.

Scott Hanselman: And then it switched to code first. I think I'm with you. I'm still back in the contract first world.

Matt Davis: Well, the thing that bothers me the most, if you skip message contracts in Indigo, in WCF, you end up with -- everything is a method call, right?

Scott Hanselman: Yup.

Matt Davis: So, if I need to add a field, well, I just screwed myself because the next time I regen my client, all of the other stuff is going to break. I don't have as much control. If you do something

really screwy where you like change the type of a parameter, you can get some really subtle, interesting, hard to debug issues.

Scott Hanselman: Yeah.

Matt Davis: I don't know. I'm more a fan of the message contract programming, so basically all the operations are defined in terms of message contracts.

Scott Hanselman: Okay, so ASP.NET guy makes a service proxy, passes some stuff to WCF. Do you have actual physical tiers? Sometimes I think that there is a bit of confusion when people use tiers versus layers. I think of layers as logical things and tiers are physical things, so there's a physically separate box from the ASP.NET box that handles all the WCF hosting.

Matt Davis: Yes. In our deployment scenario, that's true. In development, obviously, we're not messing around with that. The tiers are both logical and physical. In deployment, they are set up for physical separation. For instance, the stack that we're going live with on our .NET system is eight servers. There are two web servers, two hub servers that handle things like billing, email alerts, all the little just eventing stuff, and then we have these two large image servers, so they have a 2 terrabyte file storer on them each that store all of the images and PDFs and all that stuff. Actually, those are also running a WCF AJAX service, which is new for 3.5. We're self-hosting a WCF AJAX service in there. If you look at the client, the way the client works, when it requests all the metadata about mail, so when you look at all your envelope images and all that stuff, it's making an AJAX request. That's all client side. It's making an AJAX request directly back to that image storer. The image storer is handling all the marshalling. That's where all the LINQ to SQL stuff that's going on for mail metadata.

Scott Hanselman: I'm an evil guy and I want to sniff that AJAX and I want to go and try to look for other people's mail.

Matt Davis: Well, go ahead. Give it a shot. It's probably not going to work.

Scott Hanselman: So, how do you handle that kind of authorization? I mean is it all based on cookies? That forms authentication ticket is kind of fundamental.



Matt Davis: We weren't able to use any of that stuff because trying to host forms authentication outside ASP.NET is kind of tricky. Like I said, we're self-hosting the WCF AJAX services over there.

Scott Hanselman: Right.

Matt Davis: So, the things that are running on the image servers, there is no IIS or there's no web server involved.

Scott Hanselman: When you say self-hosting, you're saying there's like a consular Windows service?

Matt Davis: Yes.

Scott Hanselman: Oh.

Matt Davis: There is a Windows service that actually hosts the WCF, the new web HTTP binding. It's super fast, super lightweight.

Scott Hanselman: Right.

Matt Davis: It's really nice not having IIS in the mix, but you're kind of out in the lone, dreary world there because you don't have the stuff that comes in the ASP.NET pipeline that you get so used to. You don't get your cookies. You don't get your caching. They give you a headers collection and you can stuff whatever you want, you can look whatever you want, that's it.

Scott Hanselman: Right, exactly. Here, good luck with that.

Matt Davis: Yeah. There's no built-in cookie handling. There's no nothing there.

Scott Hanselman: Have you thought about using Vista and Windows Activation Service?

Matt Davis: Yeah. Actually, I wish we had built our dev machine with Vista.

Scott Hanselman: It's kind of screaming for it, you know.

Matt Davis: Yeah.

Scott Hanselman: For that particular instance.

Matt Davis: Right. As it is right now, we don't use Windows Activation Service, so there is a console app like when of the developers are working with something, there's a console app that hosts all the other individual services.

Scott Hanselman: I see.

Matt Davis: And then we just have a few config entries that point everything to the right place. One of the other really big problems that we ran into there was that the behavior of AJAX when it comes to multiple hosts is kind of loosely documented. Everybody talks about cross domain. You can't do cross domain. It turns out you also can't do cross host, which is different than everything else. If you want to serve images or anything from a browser like through an SSL connection or not, it doesn't matter where, you can request from different hosts on the same domain and that works fine. AJAX doesn't work that way and I found that out the hard way.

Scott Hanselman: What do you mean? Can you give me some examples like www1...?

Matt Davis: Well, sure. For instance, I told you that our image and mail metadata servers are running on separate physical hardware. The original intent was that those things would just be a separate host. It would be ajax.earthclassmail.com and you would make a request to that.

Scott Hanselman: A separate subdomain within the larger domain.

Matt Davis: Right, it's just separate hosts. It shows up as a separate virtual IP. Everything is all in the Active/Active stack.

Scott Hanselman: Right.

Matt Davis: However many boxes we need, we've got load balancers sitting upfront that is just doing stupid load balancing. Everything is hunky-dory. You can't do that with AJAX.

Scott Hanselman: So, it has to go back exactly to where it came from.

Matt Davis: Yup.

Scott Hanselman: That's why people talk about having AJAX proxies basically.



Matt Davis: Yeah.

Scott Hanselman: Services that do nothing but proxy AJAX traffic.

Matt Davis: Yup, and this is my first real experience with AJAX.

Scott Hanselman: I see.

Matt Davis: I kind of poked at it and played with it a little bit before, but I never had to deal with it on any kind of a scale where I would need to do something across multiple hosts. We ended up having to do some magic on our load balancers, some layers having content redirection to say, "Anything that says /json goes to that server."

Scott Hanselman: Right. The other way would have been to have the ASP.NET servers that the page was served from basically host a proxy that did nothing but left and right, which would have been probably more of a hassle.

Matt Davis: Right. It's just a waste of bit. Why channel the bits through the ASP.NET server at all?

Scott Hanselman: Right, when the load balancer can do it.

Matt Davis: Yeah.

Scott Hanselman: So, back to the security thing. You don't have cookies. You don't have a lot of automatic stuff going on. How do you get the image server to not give back images that the user shouldn't be allowed to see?

Matt Davis: Basically, what we ended up having to do is roll a little bit of our own sessioning infrastructure that goes against the DB. One of the nice things is that we don't have security below the mailbox level. If you're a business account or if you have multiple people in your household, your Scott Hanselman mailbox, you can grant access to your mailbox to someone else to be able to snoop around and see your mail like if you're in a business and you have a secretary that you want to be able to see your mail or whatever. You want to delegate the handling of your mail to someone else in your family while you go on vacation or whatever, you can do that, but there's no security below the mailbox level. The way all that stuff gets stored, what we can do is basically

we can grant you a little token that says, "This is Scott's session," and back in the DB we know what mailboxes you have access to and we just do an access check, so whenever you make a request, I mean you still have a cookie that came through, but it's a cookie that we assign to you. So, we're not using ASP.NET sessions or anything like that to do this. We kind of had to roll our own session.

Scott Hanselman: So, the request comes in for the picture of my Domino's Pizza coupon that was sent to me. We go to the database to make sure that this is, that the ticket that I came in with is Scott's ticket. Scott's mails are these things. Is this file that just got requested part of Scott's mails?

Matt Davis: Yeah.

Scott Hanselman: Somebody goes and tries to ask for a different unique ID and tries to sneak, "I'll look at someone else's mail." It's not part of his mailbox, he can't see it.

Matt Davis: Right. They're all mailbox qualified. There's actually not a DB hit involved in any of that except for the initial access check on the mailbox, which we cache, so basically you're always going to be coming back to us probably asking for mail from Scott's mailbox except in some really strange administrator scenarios where we allow like an administrator to the have access to multiple mailboxes.

Scott Hanselman: Right, or if I'm evil.

Matt Davis: But even if you're evil, we're going to tell you to go away if you come and ask for an image that isn't in your mailbox. The way those things are stored on the file system, we know what mailbox they're stored in. There are some pointer files and things that are going on in the file system, so that we don't actually have to hit the DB for every one of those hits. Our image servers are really nice and fast on the system.

Scott Hanselman: That's nice. That way someone can't kind of cause a denial of service by virtue of the fact that an evil person asking for a lot of mail that's not his doesn't affect the database.

Matt Davis: Correct because we also cache the negatives.

Scott Hanselman: Nice.



Matt Davis: If you come to us and try to hit some of other mailbox, we're going to say no. Usually that gets logged as well.

Scott Hanselman: Oh, yeah, right. Of course, it's auditing, right? This is about your mail.

Matt Davis: Yes.

Scott Hanselman: Very cool. Just real briefly, you're talking about doing WPF on the client side. When people hear WPF, they think kind of transparency and animation and all sorts of stuff. Are these pretty sexy looking applications for managing mail on the frontend? Why WPF and not just WinForms?

Matt Davis: They're a lot sexier than their WinForms counterparts, but there's a fair amount of data binding and obviously we're dealing a little of image manipulation here. Some of those things are a little easier to do in WPF with the stuff that comes in the box. It's a lot easier to scale an image and get it to work the way you want. We haven't actually deployed that software yet either. We had done a couple of prototypes and test deployments of it, but the look and feel definitely went over well with the folks over in the warehouse that are running it.

Scott Hanselman: That's interesting. A lot of people think about the pretty stuff, but the first thing you said was the data binding. That's kind of like one of the cool things about WPF that I've read and I read it in Chris Sell's book.

Matt Davis: Deep data binding is the way of the future.

Scott Hanselman: Is it really?

Matt Davis: Oh, man! Binding path? It's all about binding path.

Scott Hanselman: It's all about binding path?

Matt Davis: You can dot into an object as you want for data binding. You try to do that in WinForms and it's kind of a pain.

Scott Hanselman: Really?

Matt Davis: Yeah.

Scott Hanselman: It sounds like you're breaking the law of Demeter at that point if you can go three or four deep into an object.

Matt Davis: We don't usually go that far.

Scott Hanselman: Cool. Well, Matt, I really appreciate you taking the time to talk to us here on Hanselminutes. Maybe we'll come by sometime with a video camera. We'll do some actual demos, maybe do kind of a screen cast or something with some of the cool stuffs we're doing. So, you can go live with beta 2.0 if you wanted to, right?

Matt Davis: Yeah.

Scott Hanselman: You got the go live license.

Matt Davis: Yeah. That's actually our plan. Even if the 2008 RTMs before we actually hit it and just because we've done all of the testing in beta 2.0, we'll probably end up deploying on beta 2.0 rather than do a quick release within a few weeks after we've actually had a chance to figure out what change between beta 2.0 and RTM and I'm looking forward to it.

Scott Hanselman: Cool. All right. We'll see you next week on Hanselminutes.