



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 #134

October 16, 2008

StackOverflow uses ASP.NET MVC - Jeff Atwood and his technical team

Scott chats with Jeff Atwood of CodingHorror.com and most recently, StackOverflow.com. Jeff and Joel Spolsky and their technical team have created a new class of application using ASP.NET MVC. What works, what doesn't, and how did it all go down?

(Transcription services provided by [PWOP Productions](http://PWOPProductions.com))



Our Sponsors

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

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

NET 
<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 #134, recorded live Wednesday, October 8, 2008. 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 Jeff Atwood, Geoff Dalgas, and Jarrod Dixon from stackoverflow.com.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes and I have the pleasure to sit down today with Jeff Atwood, Geoff Dalgas, and Jarrod Dixon from the currently epic and rising stackoverflow.com. Thanks guys for taking the time to chat with me today.

Geoff Dalgas: Sure.

Jeff Atwood: You're welcome.

Jarrod Dixon: My pleasure.

Scott Hanselman: So how is it going, StackOverflow, has it cut as big and wonderful as you thought it would?

Jeff Atwood: I think we're still recovering from the idea that it's actually working like it hasn't fallen over because we did like get "slashdotted" although Slashdot isn't quite the monster they used to be.

Scott Hanselman: isn't that funny? I got slashdotted twice last month and both times didn't notice it not only because I don't read Slashdot but because it didn't do a change of traffic on my site really.

Jeff Atwood: Right. Yeah, it's not quite like the monster it used to be but there was a lot of traffic on the first day and in fact on the first day is still the peak we haven't gotten back to.

Scott Hanselman: Really?

Jeff Atwood: Well, because there is just so much publicity around on the first day but we're still inching up. There's a lot of traffic so if we're going to fall over, we would have definitely fallen over on the first day and I think I'm just impressed that we didn't frankly.

Scott Hanselman: That's awesome. So let's back up and for those who haven't gone to stackoverflow.com, what's the concept?

Jeff Atwood: So the concept of StackOverflow is something that me and Joel Spolsky -- sort of a joint venture that we put together and it's based really on a couple of things. One of the largest reasons we went after this was we felt like there is this site whose name rhymes with *expert sex change* that we think it's kind of a little bit evil in the way that they approach the world in that they're letting people ask and answer these questions but they have this sort of slimy way that they present the information and it's like they try to trick you every time you go there and it does feels very much like used car salesman.

Scott Hanselman: You Google for stuff, you find them as the answer and you can see part of the answer in the Google index, then you click, and then they lie and they say pass.

Jeff Atwood: Right. Well, you can actually scroll down. That's the trick of that site. If you scroll all the way down then the information is in back there and there are some although we're also finding that that particular site tends to be a little more IT focused. There are some programming questions there and you will get hit like I do get organic hits on them and it's the sort of thing wherever you see the links and the result, you sort of grown to yourself like, oh great. There might be something there but clicking on it is just an exercise of futility most of the time. It makes you sort of hate yourself when you go there and we kind of wanted a site like that for programmers where when you there you didn't hate yourself and maybe you actually even enjoyed going there, and the other thing driving it is I feel like there's a lot of really talented programmers who don't blog and they don't really have an online presence like, say Scott,, you and I do. You know, where we talk about, we're noisy, we're noisy, but a lot of programmers are very quiet and yet they have really great information and we want to sort of unlock that and make it really easy for them to come in and participate and ask, answer questions in an extremely low friction way and by that I mean you don't really have to log in. So it's a little bit like with PDN in that regard.

Scott Hanselman: You're trying to create the developer's third place.

Jeff Atwood: Maybe, maybe.

Scott Hanselman: You know the theory of the third place, right.

Jeff Atwood: Sure, sure.

Scott Hanselman: Home, work, and then either church or a bar, this is their third place.

Jeff Atwood: I sort of got tired of telling people to blog too. I was just like, okay, you should just start to blog and you realize eventually that although that has worked really well for you and I and



a bunch of other people, it's really not just something everybody is going to pick up. It's not really something that's in everybody's DNA that's actually gone and start to blog and keep writing every week or everyday but there's other way they can participate and StackOverflow is kind of like that. I really do view StackOverflow as my CodingHorror 2.0. It's like CodingHorror where instead of being a commoner on my post, you can own it. You can go and ask a question that you own just like your little blog post and it can be maintained. You now have the aspects of Wikipedia, it has aspects of voting systems like Digg and Reddit where you can vote things up and down and we're trying to synthesize all these things together and it's sort of the end result of my experience online of sort of a place I would like to participate in where I'm just a peer, I'm not like the guy who owns it, you're on my blog. It's like something we collaboratively build together, that was sort of the vision.

Scott Hanselman: It's like a community garden.

Jeff Atwood: Yeah.

Scott Hanselman: Where I've noticed that there's not a lot of chaff around the side, I mean people are tending to their questions and I like that there are a thousand different ways to find the same information and it's all where I think it would be. You click on your name, it goes to your profile. You go down, it says, oh, here's all the questions that you've ever asked. You go to see what you've never voted on. I mean, you're tracking everything. You throw nothing away.

Jeff Atwood: Right, that's right.

Scott Hanselman: You throw nothing away because it's all valuable.

Jeff Atwood: Right. This is a programmer's site. I mean we built it like programmers love data, programmers love nothing more than going to Task Manager and I have this sort of running joke of what do programmers do when they encounter a new program? They go to the options dialogue and they see what's available. That's like the first thing I do. I only go to the options dialogue. That's how I judge a program. Like when I installed Chrome? I was just blown away by the fact that my browser had like a Task Manager. I was so sold. The minute I saw that, I said, "Oh my God, a Task Manager." They totally get it right? And StackOverflow is a site for programmers so we have lots and lots of metrics and like ways that you can drill down the data because that's fun for programmers. We love that crap, that's what we do. So it's very much for specific audience. One way we want to avoid sort of the Yahoo Answers problem and we did a lot of research on related Q and A sites before we started this and one pitfall we found with sites like this is the more broad you go with audience, the worse it gets so we're trying to stay

very laser focused to software developers. We're not trying to meet everyone's needs in the world like all the IT people, you know, internal WIKI and things like that. We're just trying to meet the average programmers' needs on the site and stay focused on that audience.

Scott Hanselman: Don't you think it's getting a little sloppy on the homepage though because you've got Ruby guys next to Perl guys next to mainframe programmers and I know I don't want to answer those questions, I don't want to read those questions. I want a homepage that's customized.

Jeff Atwood: Well, we are working on customization and I think that's becoming an increasing issue now with the sites getting more and more people attracted to it, but honestly, part of it I think it's a little bit of an attraction. One of the things I like about my blog getting harsh is that it tends to attract people that program a variety of different things. It's not initially like a Microsoft centric site like a Microsoft developer, here's my Microsoft tools, but it attracted people from really all walks of life in programming and I love that aspect of it and there's a little bit of that we're really shooting for in StackOverflow. We're trying to avoid sort of being the ghetto of like, oh, this is the SQL server ghetto or this is the C# ghetto where we just do C# and if you post a VB question, we're going to make fun of you, or it's not the correct language. So I think part of it is really rubbing shoulders with programmers from other walks of life, not totally because obviously I can't answer Perl questions. I don't really know Perl

Scott Hanselman: That's interesting. How about -- here is a thought. I could see you getting into the *ghettoization* of StackOverflow if you simply let me as a user come on and say I only want see ASP.NET stuff here, and then, but if you have an option, if you have a slider bar that lets me select how much leakage I wanted, the leakage point of it, that's user interface word --- you know what I'm saying?

Jeff Atwood: Right.

Scott Hanselman: I want only ASP.NET but I will have 10% leakage so let other stuff randomly leak into my homepage.

Jeff Atwood: Right.

Scott Hanselman: That would be cool.

Jeff Atwood: Right. Now, definitely customization is sort of the great undiscovered country for us and we're getting deeply into it now because we realize that and particularly with regards to answering questions like it's fun for me to browse these questions and read them sometimes and it's surprising how people would just come up with a really interesting topics across a variety of different



languages that are intriguing like it's kind of just fun to read what people are posting on some level, but in terms of actually answering stuff, like I said I don't know Perl so if you have a Perl question, I may, read it -- it might be fun but I'm not going to be able to answer it so in terms of showing unanswered questions, that's the first area we're going to get into really hardcore customization and where we're going to try to show you only things that you have a track record of participating in like if you posted 50 answers to ASP.NET questions, then, well, that's what we're going to show you. We're going to show you things dealing with the ASP.NET in terms of unanswered and then that's going to start bleeding over into other areas of the site from any of the answered page. It's like that something that Jeff is working on at the moment.

Scott Hanselman: It can be really interesting to see what the longevity is, because with games like this and games like Killed Wars, there's the initial excitement of frantic leveling up and then the *plateauing* and then there's depression. How to push this pass that, that will let you know I'm at level 70, I've got all the gold badges...

Jeff Atwood: Right.

Scott Hanselman: Then how do you keep them interested and the way you keep them interested is it has to leak into their lives because they are on Google and they start searching for it and when you start showing up as some large percentage of answers, then it gets interesting and then StackOverflow really sneaks into their lives from other direction.

Jeff Atwood: Right. Well, one thing I found in participating in sort of online forums was that those were very much a game anyway. I mean, there's no level, there's no score, but I realized at some level almost everything you do online is kind of like a game. Maybe this is just a larger metaphor for life but there's some aspect of were you playing this game and we're just making that a little more explicit and to be clear, everything that's in StackOverflow that's game like in terms of reputations, scores, and badges and things like that are really there to drive good behavior. They're not in there in arbitrary way where it's okay, we're just going to give you points and you're going to earn points. You get points. Here's an example, so how do you exactly get reputation? It's very much like Google page rank system in that you don't get reputation yourself. Other people give it to you. That's the only way to get reputation system with very, very minor exceptions. So for you to get reputation, other people have to like your stuff. It has to be validated by other people; whereas in a traditional multiplayer game like World of Warcraft, you can level up by just killing random things on the site. If you just sit there and kill rats for days and days and days and there's a point diminished or returned

but eventually you will level up. On our site, you can't really kill like Warcraft or whatever. Other people have to like what you're doing and click up on it. It actually gives you reputation so it's sort of very much a positive feedback driven by the community, the other people liking what you're posting and finding it helpful and interesting and useful. So we hope, it is game like but I do want to be clear that the reputation comes from other people and ultimately what we're trying to do is stand shoulder-to-shoulder with your peers and post cool and interesting things that they react positively to.

Scott Hanselman: Cool. Let's take the conversation into a more technical direction and see if we can hear from Jeff and Jarrod. So how long did it take you to build this and what did you build it in? Maybe one of the guys could share the laundry list of technology.

Jeff Atwood: Yeah, in fact I think Jarrod take that because Jarrod did so much really great work initially getting the project up and running. So Jarrod, do you want to talk about just starting out and how we did it?

Jarrold Dixon: Sure. Actually, one of you came to me and said why don't we try this new ASP.NET MVC framework like the first preview was out and I've been working and doing a lot with Ruby on Rail just personally and I really want to use that in MVC framework. So I started out going in that direction and after using web forms for six, seven years, I've really want change and it's like a breath of fresh air working with the ASP.NET MVC framework. So we put this on the front-end with that and at the back-end we use SQL server 2005 and that was nice to use. I've been using Oracle for so long so I guess it took us about three to four months to code most of the site and working with the different preview releases, that's been a real good challenge.

Scott Hanselman: When you consider this, how did the design process go? Jeff, did you visualize this in your head and then draw it on a napkin and then you got a designer? Is one of you three a designer?

Jeff Atwood: No. Design was one of the big challenges. So Joel and I sort of had ideas. Joel, I think, had his .NET discussion group *Joel on Software* discussion boards which occasionally I get organic hits of those as well when I'm searching for something on Google so I was aware of those. I wasn't really participating in them but they're extremely barebones like by design and Joel has a whole post about that but Joel and I had some clear thoughts about how you build the community and sort of the ways you combine these features together to make it interesting and useful. So I think that was a lot of the initial that Joel and I were just going back and forth and then Jarrod and I did a ton of research. We spent a week going to just like every Q and A site on the internet



like almost literally. Jarrod and Joel talked a little bit about that.

Jarrold Dixon: Yeah, whenever we would send each other links of some of the hilarious questions. Jeff had mentioned earlier he had to keep a ledger focused on software developers because like for instance Yahoo Answers that people are asking crazy questions like how is the bat deformed, you know.

Jeff Atwood: It's really amazing. Yahoo Answers is disturbing how weird it gets so rapidly and because the topics are so broad. You can ask literally anything and when you can ask anything, it's just sort of craziness ensues and the example I use is YouTube where if you have a really popular YouTube video, the comments are just going to be crap. But here is the thing, if you find a YouTube video that's really narrow, you'll actually find some really useful comments there. It's surprising. There's like two classes of usage around YouTube. One is for the generic, you know look at a woman, a scantily clad woman dancing video and there's like a super narrow video that maybe only 10,000 people in the world care about and those have really good comments and our thought was, hey, let's have a world of incredibly narrow technical problems that only 2,000 people in the world would care about and that's sort of how you get the signal to noise ratio up because the signal is so narrow. That was the sort of thing we found there and it's amazing how many of these sites there are. There's AnswerBag, WikiAnswers and Jarrod and I were finding all these sites. We have no idea...

Scott Hanselman: Yeah, I've never heard of these things. It's amazing, I've never heard of these sites. Are these just really boutique type of things?

Jeff Atwood: No. I think they positioned themselves as the world's interface for asking questions and a lot of them have a pretty large community and we agree, we were surprised. We have never heard of these sites but we were trying to get ideas about how do you run a Q and A site and how are people currently doing and the sort of things that are running into. You know, that was a good research phase and that lasted about two weeks or two weeks and the rest, it was just taking our collective experience. Certainly, we've been like MSDN forums and other sort of programming hangouts that there are.

Scott Hanselman: Sure. One of the things that really strikes me, and maybe it's just me, about the site was that somebody made a conscious decision to use all text for everything so that I hover over virtually any element on the screen and I get a little tool tip.

Jeff Atwood: Right.

Scott Hanselman: That just seemed really both obvious but also really clever because it gives you extra information anytime you pause and go, oh, that's what that is, because the tool tip pops up. Was that some dogma that came from on high?

Jeff Atwood: That really came out of the feedback. So one thing I did early on was I emailed everybody I could think of that I sort of respected in terms of they had no sites like this or they had done things on the internet that were really impressive, that they have sort of created a site doing something that I like and I said, you know, please take a look at this and I said email me directly and I took that feedback extremely seriously because these are people who have a track record of building things from the internet which I'm really drawn, I mean I have my blog and stuff but I haven't really build a site and put it out there and one of the key pieces of feedback I got back was use tons of tool tips tactics and I really took that hard and I tried to do in tons of places that are out there in the site. So really a lot of what we do is really driven by feedback, not everything but certainly we try to sort of fold in all the good ideas that we're getting from people also that we see on other sites from the internet.

Scott Hanselman: Hi, this is Scott Hanselman with a word from our sponsor. Do you know how to build a web 2.0 AJAX application with web 1.0 components? You really can't. You want to do the next generation web applications? You'll need next generation components just like the ones our friends at Telerik have got, their RadControls for ASP AJAX. It is a huge pack of web controls built on top of ASP.NET AJAX that will add previously impossible performance in your activity to your next project. The new controls mirror the AJAX API from ASP.NET so development is really straightforward. The client scripts are shared so loading time is not a problem. You just set a couple of properties and you'll be able to automatically bind the web services for a really efficient operation. The new RadEditor from ASP.NET AJAX Telerik loads up to four times faster than before; and the new RadGrid handles thousands of records in just milliseconds, but as always, it's best to try it for yourself. You can visit telerik.com/aspnetajax and download a trial. Thanks a lot.

Now I'm looking at the homepage here and I'm seeing recent questions, recent tags, recent badges, and then within each question, you've got how long ago it happened, who did it, then statistics on that particular individual. What does the shape of the data look like on the homepage? How many queries are happening here? Are there dozen little queries or one epic query with a lot of -- what is the shape of this data?

Jeff Atwood: We're doing a lot of queries. We're using LINQ to SQL as our underlying proof, DAL like as you would say. I tend to lean heavily



towards the database as model camp. I know this is heresy to other people but...

Scott Hanselman: Oh yeah as well as not learning C.

Jeff Atwood: Yeah, yeah, exactly, not learning C. But LINQ to SQL is nice in that regard because it's very lightweight. It essentially almost like having raw SQL in an application and in a lot of places we do still have raw SQL in our app because I'm a little bit old school about that but I love Link, I want to be clear. LINQ is super...

Scott Hanselman: Really, really. Are you saying that you break the fourth wall and you just go right down and you pull out ADO.NET in some spots?

Jeff Atwood: Well, we use the LINQ context so DB.Execute come in which is a link, accepted LINQ way of...

Scott Hanselman: Yeah, but underneath it's still going right to the database.

Jeff Atwood: Yeah, in some places we are but in general we're going to LINQ so there's very little difference. The homepage for example, we're doing some group inquiries but we also found that the way MVC works, it can be a little bit of a challenge to figure out sort of where the data access is occurring because you'll build the model, you know, model view controller and your model will have some property that will makes it go get more data and it can be kind of subtle sometimes about when you have a view accessing one of these properties on the model and then that triggers a whole bunch of queries so then effectively your view is querying the database. You really think about what's happening which depending on again on how dogmatic you want to be at .MVC, you're not really supposed to do that. The viewers must be really dumb and just dump data on the page in a specific format but it can be subtle, figure out okay, if I access this property, bam, 50 queries occur.

Scott Hanselman: Yeah.

Jeff Atwood: And Jeff, do you want to talk about some of the optimization we did around because I think caching was our biggest challenge, honestly, with MVC because I think what we run into was you have controllers, we kind of wanted sub-controllers because we wanted a way to sort of break up the data so that we can actually cache for the sub-elements of the page but not the whole page. So Jeff, do you want to talk about that at all?

Geoff Dalgas: Sure yeah. I guess I'm a little embarrassed about our implementation of it as well.

Scott Hanselman: Don't feel embarrass because it's not like you're on stage trying to be purist and justify your desire...

Geoff Dalgas: I guess.

Scott Hanselman: Because ultimately you guys need to remember that success is ultimately the best metric.

Geoff Dalgas: Yeah, very good point.

Scott Hanselman: And you guys are doing a good job and you balance pragmatism with purity so it may not be academically correct but let's talk about it in a frank way without feeling that you're being judged and then after you open your soul to me, then I'll judge you, but until then...

Geoff Dalgas: And so we're using some of the MVC contribs to get basically our views call back into our controllers in order to leverage caching...

Scott Hanselman: Call back into it via what? Like they literally -- typically you pass from your controller to your view, you pass a view model.

Geoff Dalgas: Right. There's a method called render action that allows...

Scott Hanselman: Right.

Geoff Dalgas: I think it's gone in the latest version 5.0 of the MVC framework so I actually had to find it in the contribs where I'm using render action to call back into the home controller to basically have the view populate all of the data and by doing that I'm then able to use the values that come back from the HTML to cache that and we're able to cache regions of the page instead of just the entire page.

Scott Hanselman: Is the caching done as an action filter on the controller?

Geoff Dalgas: We are using the action filters for the caching to occur but we didn't want to cache like the entire homepage for X number of seconds. We used the ability to cache, let's say if you're looking at the recent tags on the homepage, we wanted the recent tag section in a page to be cache for five minutes or for the next interval, but leaving one of the rest of the page to be able to refresh over one minute interval. So we have a view that shows just the default page and then that view calls back into the controllers to get the information that cache on different intervals. That makes sense.

Scott Hanselman: Right, okay. So then the home controller has a recent tag thing that has an attribute on it that describes an attribute output cache in five minutes.



Geoff Dalgas: That's right.

Scott Hanselman: So you're caching them at the render action level.

Geoff Dalgas: Right.

Scott Hanselman: Okay. Are you running all this on -- how many boxes do you have?

Geoff Dalgas: In this case, we just have one server that's acting as both our database and web server and we run pretty hot. We're running about 40% to 50% CPU at times. That's kind of during our really heavy loads so...

Scott Hanselman: What kind of box is this?

Geoff Dalgas: It's a quad core, I believe Zion, is that right Jeff?

Jeff Atwood: Yeah, actually it's eight-core.

Geoff Dalgas: Eight core exactly.

Jeff Atwood: Dual quad. So we have 4 gigabytes of memory and then 8 real CPU's in there through a single processor...

Scott Hanselman: What's your memory...?

Geoff Dalgas: Right.

Scott Hanselman: You're running 40% on average when you're running hot on all eight cores.

Geoff Dalgas: Right.

Jeff Atwood: Well, 40% accumulative, so that would mean we're doing 40% load.

Scott Hanselman: And what is your memory pressure like with only 4 gigs?

Jeff Atwood: It varies. We've improved our querying and one thing we found, I know it's sort of a running joke in computer science, but like all your performance is in the database, I mean it's just ridiculous how much everything you do is termed at how fast you get data out of the database and that's why also the caching turned out to be obviously super critical for us. I just want to talk about just a couple of issues that we have during development and one was the current version of MVC is not very cache friendly because it doesn't really let you break up controllers into sub-controllers because really, as Jeff was describing, you have a page that's really a bunch of sub-items even though it's one controller action and you want to cache different parts of that controller differently so you really want to compose your controllers into sub-controllers and then each sub-controller will have a view associated with it that

would have its own cache and stuff. That will be a little bit of a challenge because for our site of our scale, we have to cache like crazy otherwise we just fall over. We can't do hundreds of queries for every user like every second that they hit the page. It's just like web designing when I want and then beyond that, beyond the caching with MVC, I was really actually very disappointed in the way SQL server works out of the box, SQL server 2005. We had huge issues around deadlocks with what I feel like is not an unusual query load for a website e.g. 98% Read and 2% Write as that we were regularly blocking and deadlocking like multiple pounds per hour and it turns out, and I had a blog post about this but like I said I'm still kind of disappointed that they, out of the box, made a decision that we had to switch to this Read committed view or this transaction level which gives you -- it's not like no lock, no lock is Read uncommitted. I know you know about that because you blog about that before, but this is one level up from that and once we switch to that, it had been totally smooth sailing but until we did that, it's like pulling out hair.

Scott Hanselman: Do you feel there's a little bit of backlash against that post? This is a tough question since we're such wonderful friends but there were some people who said that that post show you really didn't even understand the issue around deadlocking. Do you disagree?

Jeff Atwood: Well, I think that there are a lot of people that don't actually read the entirety of what I write like they read like a couple of paragraphs and assume that, oh, he doesn't know what he is talking about, but they don't sort of read the end because I think reading that post, if you actually read everything I wrote and get to the end of it, I'm just painting the entire picture of, you know, here's the problem...

Scott Hanselman: Right.

Jeff Atwood: Here's one thing you could do and then here's actually what you should do and in the what you could do was no lock. We talked about that and honestly I've talked to people that have worked as DBA's and just tell people like yeah, use no lock and everything. it's like take two aspirin and call me in the morning. There are DBA's out there who do this and I think that just because they've learned that in most you don't you have locking problems and for a lot of queries does it really matter like sampling and pulling back a user profile, does it really matter if the profile is a little bit out of date at the time that I get it. I'm not sure that it does.

Scott Hanselman: Right. I'm hearing. You said that you wish SQL server have a more realistic awareness of how the world typically works and have typical defaults is what you're saying.



Jeff Atwood: Well, it turns out that this Read committed transaction level; I think it's the default in like Oracle and stuff and people have left comments on my blog post, wow, that's not the default. They were really shocked that this was not the default and I think we were too because I do not view our query load as unusual, certainly not for websites. So out of the box, SQL server is just really ill configured to build a website. I mean, that's really the bottom line. That was the point of that posting. Hey, look, out of the box SQL server needs to be adjusted to work with sort of typical website loads and once you do that, it's great, I mean we haven't had any problems since then.

Scott Hanselman: It would be nice if there's a profile option like you could tell SQL server here's how I intend to use you, set yourself up appropriately.

Jeff Atwood: Well, right and I have sort of heretical view on this anyway that I think that the database should be figuring out a lot of stuff like you should index this column, like I think the database should be doing lot of stuff but not automatically.

Scott Hanselman: Yeah, that's just cookie talk. Yeah, yeah, I mean it does learn a lot. I mean SQL server and particularly SQL server 2008 got really, really smart on query plans which interestingly I've said a few heretical things that got me in trouble like I believe that the day of stored procedures is probably over and the kinds of...

Jeff Atwood: Oh, I've said that thing years and years and years.

Scott Hanselman: Yeah, okay. You're a progressive heretic.

Jeff Atwood: I am totally.

Scott Hanselman: You're ahead of your time.

Jeff Atwood: And you know what, Geoff Dalgus was telling me that Rob Connery added an Atwood message to subsonic that actually extends for all SQL. Supposedly that was mentioned on one of the Hanselminutes podcast.

Scott Hanselman: Yeah, he and I talked about that last week.

Jeff Atwood: I sort of mention that but I do have affinity. I kind of like SQL, I will say that. I know Rob Connery kind of hates it but I love it like I think it's like that to me is database assembly language because the database is so, so important to your performance. I mean it's like V thing that determines your performance. So if you don't have like a real lock on how you build SQL and like the performance character sits at the SQL you're building, I mean

you're really just dead in the water. You're going to build an app that's going to be incredibly slow. So I view it as incredibly mission critical to deeply understand the actual SQL that you're emitting and I don't really like the sort of -- LINQ does this very well. To be clear, LINQ does an outstanding job of this but still I want to see exactly what SQL is being generated and understand exactly why it's performing the way it's performing because when you look at StackOverflow, 98% database junk.

Scott Hanselman: It's fast. I mean, it's exquisitely fast and I thought it was pretty cool when Obie Fernandez came on and asked if it was written in Rails. That was pretty awesome.

Jeff Atwood: Yeah, that was a great question.

Scott Hanselman: Let me push back on one thing though because I'm hearing that you're a pragmatist and you're a pure pragmatist. You know, make it work, do it right, get out of my way, let's go, go, go. But I'm shocked that you have your SQL server running on the same machine as your web server.

Jeff Atwood: Well, this gives us a really obvious upgrade pass. I mean this is one way of looking at this; it's that we can...

Scott Hanselman: We have to go live with the SQL server on. It's not behind the firewalls, it's not at its own -- you know, it needs to be out of the DMZ and behind the second firewall. I mean, I worked in finance for years so maybe I'm just paranoid but that's just the surface area for attack, just doubled.

Jeff Atwood: Well, it's behind the Windows firewall stuff and there is really no way to get to it unless you can remote desktop in like we do.

Scott Hanselman: You see, now you've just said that that you remote desktop in your production machine which adds a new attack record so that means that now I know you have ports open for RDP.

Jeff Atwood: Well, what are you going to do? I mean, people are going to attack you with whatever is available. Our passwords are pretty good.

Scott Hanselman: Well, so that's not the point. That's like saying the lock on my door is pretty good, but you know, most secure house is one without doors.

Jeff Atwood: Right.

Scott Hanselman: And even more secure if I can't find the house.



Jeff Atwood: Right. Well, like I said we have obvious upgrade path which is to move the database to buy another server from Crystal Tech, that's our house and shift the database so we now immediately cut our load in half on that server.

Scott Hanselman: Do you think that that's really that simple, it's half? That half of your CPU's being used by the SQL server and the other half is being used by the SQL server?

Jeff Atwood: Yeah, it's pretty much an even split. I mean, we watched Task Manager pretty closely and assuming the processes have been up in the same length of time which isn't always true because the worker process tends to get shut down for various reasons but when they're up for the same amount of time, that's actually shocking, how even CPU's that actually is between the two at the moment.

Scott Hanselman: How far can you take this? I mean have you done capacity planning? Any form of capacity planning?

Jeff Atwood: This is Stack over-footing. We don't really do any planning. We just kind of just do stuff and just see what happens.

Scott Hanselman: Yeah, I'm just a stodgy old dude because it's like when you go to a bank, how many people are signing up for online accounts and how often is it? What does the trend line look like? What does the trend line look going backwards? How will it be exponentially geometric in 10 years or 5 years or even 6 months? What's the capacity like? So you're just going to buy hardware when it starts to get to 60% and then you'll just figure out from that.

Jeff Atwood: We've been keeping a really close arm in the database. In other words, watching for queries that start to become painful.

Scott Hanselman: Using the profiler?

Jeff Atwood: We love the profiler, by the way, like my new best friend.

Scott Hanselman: Are you profiling the database in production?

Jeff Atwood: Oh yeah, absolutely. I test the database all the time and just look at the grades and see what's going through them and what's slow.

Scott Hanselman: And are you concern about database fragmentation? Do you do anything like that? Have you figured out if your tables break on a page boundary? Do you think about that stuff?

Jeff Atwood: Not currently. I mean, most of the thinking I do is around this query takes few hundred millisecond and needs to take 50 so then I

just take it apart in the query analyzer and try to figure out how to make it faster.

Scott Hanselman: What are all your biggest tables? Millions...?

Jeff Atwood: Jarrod, Jarrod, do you want to talk a little bit about...? Jarrod did a lot of research initially looking at how Wikipedia stores data and Jarrod came up with really the initial design. It's been pretty solid, there have been some tweaks. But Jarrod, do you want to talk about that?

Jarrod Dixon: I guess our largest table will be where we store the actual text with every revision that goes in. Just said at StackOverflow, there's an element of WIKIs that you can edit any questioner, revise its tag. I would say we have, I mean it's not in the millions of records, as of now not yet, but I think we have like 400,000 records right now in our revision table.

Scott Hanselman: You know the state of fragmentation of the SQL server?

Jarrod Dixon: No.

Scott Hanselman: No, I mean just fragmentation. We're talking about like what's going on in the internal data structure of SQL server.

Jarrod Dixon: Right but no, we haven't look at that yet.

Scott Hanselman: How many tables are there?

Jarrod Dixon: Off the top of my head, I'm not at my computer. Let's say anywhere like 15, 16 tables. I mean, it's not a large schema by any means.

Scott Hanselman: So a lot of the stuff, there's little, tiny database updates that happen when you vote something up or you get a badge or you whatever, this is all just inter-walking from it kind of stuff plus one, all these different values, tiny, little...?

Jarrod Dixon: Looking for instance with voting. What we actually do is store record every vote that happens, you know, what happens to, so that we can recreate like for any given user, their reputation we can actually have a path of everything that happen to get that user that score. In certain cases we do have this straight increment but most of the increments are all sort of d-normalize fields. We can always get back and calculate how that value happens.

Scott Hanselman: When you said get back, are you saying that you calculate and store or you just calculate?



Jarrold Dixon: Like if I voted somebody's post, we would store a record and vote table like this person outmoded giving users post, then we would increment their score on the -- or their refusing their reputation on their user accounts on that record just so we have sort of a -- we don't have to calculate that reputation score every single, you know, when a user gets...

Scott Hanselman: Okay. So you calculate it and store it.

Jarrold Dixon: Right.

Scott Hanselman: But you're saying if you decided to zero out the reputation column, you could get it all back.

Jarrold Dixon: Exactly.

Scott Hanselman: Ah okay, okay. Did you think that, give any sense of how big this will get before and what will break first, will it be SQL server, will it be the memory on the SQL server, will it be your cache, does your cache try to cache everything or if you have 10 times more content, it tend to have more traffic, would your cache have to be tuned to cache less?

Jarrold Dixon: Yes, we definitely need more memory because server is running pretty much -- Jeff, what's it running at right now?

Jeff Atwood: We saw some disk cache left. It's not actually that overloaded at the moment but I think it's going to follow very traditional scaling back there. I think moving the database to its own server did vagaries of how our host work. It's actually cheaper to buy a second server than it is to upgrade memory in the first one. So that effectively doubles our memory space because then the database is on its own server with four gigabytes of memory. The web server has four gigabytes of memory and I got to tell you, Scott, we are abusing HTTP, the cache, the ASP.NET HTTP cache. We're using the heck out of the thing, man.

Scott Hanselman: Are you using kernel caching or you're just using this web.cache?

Jeff Atwood: Well, we're using really a mixture where a camera using sort of the attribute level of the caching, but the trick there is everyone of our pages has the user at the top so doing a whole page cache only really works well for us if you're anonymous in which case you're not logged in meaning the page is identical for you and every other anonymous user.

Scott Hanselman: I see.

Jeff Atwood: Yeah.

Scott Hanselman: So when you say HTTP caching, you're talking about like e-tags and client side caching.

Jeff Atwood: A little bit but we're also stuffing stuff in the cache like...

Scott Hanselman: I'm confused. There are basically three levels of caching. There is user mode caching which is like using the cache in ASP.NET. There is kernel mode caching which is like using htp.sys to see if you can turn a website page around without even hitting IIS.

Jeff Atwood: Right.

Scott Hanselman: And then there's whatever caching the client does.

Jeff Atwood: Right.

Scott Hanselman: So which are you using and in what way?

Jeff Atwood: We are exploiting some caching in terms of just having correct HTTP headers and things like that and compression and things like that but HTTP, the built-in ASP.NET cache is like our new best friend and one of my favorite new technique is not even really new or very clever. It's we're taking a lot of our HTML fragments that we're rendering and then we're zipping them and then putting them in the cache because we have so much CPU time, not as zip is a computational expensive operation anymore but that sort of increases by 10 the size of your cache because you're not putting HTML in, you're putting GZIPPed binary block...

Scott Hanselman: Really? You're using like system.io.compression?

Jeff Atwood: Yeah.

Scott Hanselman: And you're just taking, how much? Like 30K and zipping it?

Jeff Atwood: Yup, yup, we do that because, you know, they bring how many tabs you have on page; therefore, there's combinatorial explosion of the cache. Even for an anonymous user, there's say three or four tabs per page that the user could go to but that means we don't store one copy of the page, we store four copies depending what tab you're on, question for example, so this way each one of those tab is like one-tenth the size of what would normally be in the cache. So we can sit a lot more of them in without putting pressure on them and we have so much CPU, it's like comical.

Scott Hanselman: Did you take system.web.cache and wrap it and make a GZIP



cache that zips and unzips as you go in and out of it or did you do the zipping on the outside?

Jeff Atwood: I did the zipping on the outside but I can tell you, that would be an awesome addition to the core of the .NET stuff. it's just zip everything that goes in.

Scott Hanselman: Are you going to open source any of your stuff, I mean, send me that and I'll make a GZIP cache.

Jeff Atwood: And I will make, I think, a good blog entry.

Scott Hanselman: Yeah, that is interesting. Yeah, I'm familiar with the combinatorics especially it's really interesting if you did yourself in multiple languages like one of the biggest and worse in most lame bugs and dust blog to this day remains that sometimes on some pages where I do output caching but I forget to add in user language as a combinatorics like if a German guy is the first guy that hits that page, then it gets cache in German for everybody no matter what language this be.

Jeff Atwood: Right. So that's how many languages are there, like hundreds.

Scott Hanselman: In the world?

Jeff Atwood: Well, I mean that your site supports, that your site would actually localize...

Scott Hanselman: Probably 26.

Jeff Atwood: Twenty-six, so in every page it's times 26, right.

Scott Hanselman: Yeah, exactly.

Jeff Atwood: Yeah, it gets scary really quickly that's why they're zipping it. It's so awesome because you can sit literally 10 times as much crap in memory.

Scott Hanselman: You know, my initial reaction to that is one of like to stain and almost a little bit of nausea because it just didn't feel right.

Jeff Atwood: That means we're doing it right. That's good, that's what I like.

Scott Hanselman: It means it's really clever though, I don't know. I maybe really interested to hear from the listeners what they think about that. I think that you can get into like Micro-Perf and decide how much CPU that's going to take but ultimately CPU is going to faster than the amount of memory that you could buy. You can't buy 10 times more memory without getting into big money.

Jeff Atwood: No, it's not true and we have this stupid amount of CPU. It's like even on day one...

Scott Hanselman: That's pretty cool, that's pretty cool. This is IIS 7.0 and 64-bit?

Jeff Atwood: Yeah and that's true and our stack, I got to say I love our stack. I mean, I'm a Microsoft developer so it's sort of incestuous for me to even say it but our stack is really kicking ass because we're 64-bit top to bottom, 64-bit operating system and 64-bit .NET framework, 64-bit SQL server and the performance really, as long as you don't make queries that suck, this is where it gets -- it's just amazing how quickly you can struck the query that just sucks, that's just join the wrong things and takes 3 to 4 milliseconds to come back. As long as your careful with your query construction, it's amazing how fast things are performing and we haven't had to do that much tuning, mostly it's just been tweaking and paying attention to the things that are starting to get slow around the edges where places where we've done stupid things, where we didn't have the right index, where our query was really poorly constructed and things like that. So largely, we've just been removing those barriers that we put in the system to unlock the performance that's already there and a truly graph line that a lot of people from the community come in and say, oh, well, .NET really can be fast which you and I have known for years. I mean, .NET is really fast. It's just a question of how you use it and StackOverflow is a way to show that off as well and that's why I'm so excited to speak with Phil at PDC about MVC and talk about how much we like MVC and how much we really love our stack. We're not religious about it but we're proud of it.

Scott Hanselman: Are you going to start blogging *How To's* and like you've got this -- you know, one of the things that I like about my blog is, especially a couple of years ago when I was actively, actively developing was all of the war stories. You've got a huge pool of war stories and code samples just sitting in your heads now as a collective. Are you going to start getting those Best Practices out into the world?

Jeff Atwood: Well, some of our stuff isn't actually Best Practices. Some of our stuff is a little bit scary.

Scott Hanselman: Right, some of your stuff is pretty horrific, it should never see the light of day but still it works.

Jeff Atwood: Yeah. We've learned a lot chiefly around security. I've done a couple of security related entries. I'll tell you, bill the programmer a website and you learn more about security than you ever wanted to know within the first month. I know that was true for us. There were lots of exploits that we didn't fully understand and now I think we, I don't



know, we fully understand that we definitely have a better understanding of them and before we get too much deeper, I want Geoff Dalgus to talk a little bit about we've been using CruiseControl.NET and one of the things that made this really a pleasure to work on is that we have continuous integration like we can do build all the time and we deploy new versions of the site literally everyday to web servers sometimes multiple times so it's very much rapid development in the sense that we roll things out as soon as we develop them and then we get feedback immediately. So Geoff, do you want to talk a little bit about setting up CruiseControl and your experience with that?

Geoff Dalgas: Sure yeah. Well, Jeff said a lot there already but pretty much we have every check-in built process where one of use checks in and we all get an email, a built performs and the unit tests are run against that build and we all get an email whether that was successful or not and we kind of get to watch each other's work as a check in and watch the site kind of evolve and basically check our ourselves whether we're putting in good work. I know Jeff is watching us pretty closely.

Jeff Atwood: Well, we're all watching each other I would say.

Scott Hanselman: Did you say CruiseControl and continuous integration are really just opportunities to oppress you?

Geoff Dalgas: Yeah, pretty much.

Scott Hanselman: Because if you need to be rescued, we could put together a team and get you out of there.

Jeff Atwood: Yeah. So another thing, Scott, and make you sad and depress our project and probably justifiably so, our development website is on the same server as production. Do you like that?

Scott Hanselman: Oh, sweet Lord. I thought that you were supposed to be the embodiment of Best Practices because you are -- this is horrible.

Jeff Atwood: The embodiment of Best Practices is you just don't do what we do.

Scott Hanselman: Yeah. Oh my God.

Jeff Atwood: That's what I'm trying to tell people. it's just economics. I mean, we don't really want to buy a lot of servers until we really, really need them. I agree, I don't really like having development website on the same server but I guess I'm cheap at the moment because we're kind of in a start-up mode. Eventually we will have more than one server and we will do this the right way. I do see that.

Scott Hanselman: I don't even know what to say. I'm stupider for having heard this information.

Geoff Dalgas: When you work with Geoff, you realize that he's at times a little bit of a renegade but that kind of makes him a little bit more maverick.

Jeff Atwood: Geoff is really like, "Is that really going to work, do you think?" And then he gets in there and like, "Wow, okay, you actually got a homework done here. It actually works, amazing."

Scott Hanselman: What can people learn from this? I mean, you guys pulled this off in just months.

Jeff Atwood: Yeah.

Scott Hanselman: I mean, there are projects failing left and right. You either discovered a new crazy, fast and loose caffeine power way to do software or you're headed for a huge brick wall.

Jeff Atwood: The way I look at this is there are so many projects out there and not that there's anything negative about PHP developers. I don't know how many I would be offending in this particular audience but there are so many programmers that are literally just throwing crap out there on the web and just seeing what sticks. I mean, if you look at like Friendster...

Scott Hanselman: Just like you did.

Jeff Atwood: And you look at all these really classic websites, I mean these are not really well architected things. They just kind of grew into what they were. It doesn't actually pick a really good architecture to just succeed. It takes an idea that you just execute on overtime and you keep building it up. I think that's the more important factor than getting everything right. It's just having it be right enough for today and then having a commitment to improving it everyday.

Scott Hanselman: Wow. Well, I think that's probably a good place to stop because that gives a lot of people a lot of things to think about. Thanks guys, very much, for sitting down with me today and folks can check it out at stackoverflow.com and you're going to continue blogging, I hope it will be a little more often than you have been, Jeff at codinghorror.com.

Jeff Atwood: Yes, I'm coming back.

Scott Hanselman: Fantastic. All right, this has been another episode of Hanselminutes and I will see you again next week.