Dog Food Con 2017

By Andrew Hinkle

This was my first time attending [Dog Food Con](http://dogfoodcon.com/) where the topics this year followed the central theme of Artificial Intelligence (AI) and Machine Learning (ML). The conference lasted two full days with plenty of opportunities to learn from my Robot Overlords. Sure the presenters looked human, but I know an Agent when I see one.

 image from @DogFoodCon twitter page

# Fearing the Robot Overlords – Keynote – Christina Aldan

We all fear the rise of Skynet or believe we are already under its grasp. As developers we know our services are in high demand today. What about tomorrow? Will I be replaced by an AI? The short answer is yes. If you believe that you've been trained well.

AI cannot replace us, because we have emotions, creativity, and innovation. AI can be trained to perform repetitive tasks and to find patterns in big data. However, the AI must be trained, monitored, and retrained as new data becomes available and as patterns change. In the end, let AI perform the repetitive tasks for us as tools, so we can do the cooler innovations.

# Beginning your Azure Machine Learning Adventure – Lori Sites

I came into the session thinking you had to plug in some data massaged to fit your needs and run it through some algorithms. That's like saying you need some ingredients, tools, and a recipe and now you have a cake. It's not wrong, but there's a lot more to it.

At a high level ML follows the process: Define Objective > Collect Data > Prepare Data > Apply Algorithm > Train Model > Evaluate Model > Choose Model > Publish Model > Monitor, Operate, and Retrain > Rinse and Repeat. Lori did a great job going through each of these steps along with tips and traps.

Define Objective: What question are you trying to answer? Be specific. The question must be answerable. How often should you get the correct answer? Have a definition of done.

Data can come from multiple sources such as SQL, Azure Tables, CSV, etc. You better have many records. The more the merrier. You'll spend time cleaning, massaging, and filtering data. Remove nulls and outliers.

This session was packed full of data on each step of the process. I have pages of notes that I'll be passing on to our data team. I would like to see Lori run another session later on how to implement what we've learned and see it in action.

# Developing Native Mobile Apps? JUST STOP! – Patrick Toner and David Balzer

I haven't done any native mobile apps as I have little interest in writing code specific to an operating system if there is an easier way to create an app that will work for all devices. As the meeting started, I was excited.

You can make the html to behave and look like native Android and iOS. Amazing! You can use HTML5 to access native features like GPS, notifications, audio, etc. Yes sir, please tell me more. You can package the HTML5, CSS, and JS, into an app that can be uploaded into both app stores. Sweet!

I totally misread the intent of this session. With all of that said, I was expecting to dig into the code and see how it was done. Give me a tutorial; I want to do this myself! Instead, the session was tailored to convincing you to switch as an open discussion.

After some subtle probing by the audience I cut straight to it and asked for some code samples. What we learned: Use [PhoneGap](https://phonegap.com/)/[Cordova](https://cordova.apache.org/) with [framework7.io](http://framework7.io/). Unfortunately, that's the extent of what anyone who stayed to the end took away from the session.

# Top 5 Architecture Patterns – Jim Everett

I'm very interested in boning up on my Architecture knowledge, so you'll see a few more details regarding this session. Microservices seems to be the best pattern as long as you walk the fine line of proper separation of concerns to reduce chatter and keep your services performing well. However, one pattern does not rule them all as an example each one of those Microservices could implement the layered pattern and given the isolation that would work just fine. Any of these patterns could work in conjunction with the others to meet your needs.

Jim described each of the five architecture patterns at a high level, language neutral. His [ArchitecturePatterns](https://github.com/CognitiveBurden/Top5Architecture/blob/master/ArchitecturePatterns.pptx) slideshow is on github. He ranked each pattern as good or bad for six indicators.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pattern | Agility | Ease of Deployment | Testability | Performance | Scalability | Ease of Development |
| Layered | No | No | Yes | No | No | Yes |
| Event-Driven | Yes | Yes | No | Yes | Yes | No |
| Microkernel | Yes | Yes | Yes | Yes | No | No |
| Microservices | Yes | Yes | Yes | No | Yes | Yes |
| Space-Based | Yes | Yes | No | Yes | Yes | No |

He mentioned that he does not recommend using Enterprise Service Bus (ESB). I would love to learn more about his opinions on this topic.

**Layered Architecture**: It's the first pattern most of us learn: Presentation > Business > Persistence > Database. Everything is one way down and then one way back up. In this pattern if the presentation just needed to access a piece of information to display on the screen you'd have to update the entire pipeline to facilitate the request often times adding pass-through logic in the business layer to access the data.

It's a general purpose pattern that most pick up pretty quickly in their career. It's known as the sinkhole anti-pattern and is typically associated with monolithic apps. I use this pattern often to quickly organize a new app until it evolves. While not discussed, another version of this pattern involves injecting the persistence (unit of work/repositories) and database layers into the business layer making the business immutable and maybe a little less of a sinkhole.

**Event-Driven:** The Mediator pattern triggers off an event submitting information to a queue where a mediator pulls the item off the queue directing it to one or more channels, and finally processors handle the information. The Broker pattern triggers off an event submitting information to a broker directing it to one or more channels, where processors handle the information and may submit information back to the broker that may continue the cycle until the event is fully processed.

**Microkernal:** Create a core system that processes plug-in components that can enhance the system with new features. It's very extensible and product oriented, however, if the core changes then the plug-ins may need updated especially if any of the contracts change between the core and plug-in.

**Microservices:** Your components are fully decoupled into services with a single purpose. With this isolation changes can be made to a service without having to redeploy the entire application. Concepts like Domain Driven Design and Bounded Context assist in helping to know how much should be separated. Too little you have a monolith, too much and you have a performance hit from too much chatter. API REST-base topology (website), REST-base topology (fat clients), and Centralized Messaging (remote access via Message Broker) are a few patterns to follow.

**Space-Based:** AKA Cloud-Based removes the centralized DB using in-memory data grids. Processing units (PU) include your modules (DLLs) and accesses data in memory which is maintained by a data-replication engine. The PU work with a middleware that handles the messaging grid (handles requests and send them to the PU), data grid (manages data replication), processing grid (orchestration), and a deployment manager (starts/stops the processing units, etc.). This pattern can get expensive fast.

# Introducing ASP.NET Core 2.0 – Steve Smith

Steve Smith doesn't waste a breath in his sessions and are packed full of information. If you have the opportunity, watch his [PluralSight](https://app.pluralsight.com/profile/author/steve-smith) videos and sign up for his [dev tips](https://ardalis.com/tips). If you're interested in starting up ASP.NET Core, I've heard nothing but good things about his [ASP.NET Core Quick Start](http://aspnetcorequickstart.com/) program.

.NET Core 2 is modular – NuGet package based. He went through the process of installing the .NET Core 2 packages via command line for those who want to use their own IDE and also through VS2017. This is nice for cross platform development. When you use VS2017 make sure the new project page has in the dropdown ASP.NET Core 2.0 to see the new options. Steve made it very clear he does not speak from experience and did it right the first time.

The new projects include WebApi, WebApp – Razor page, and WebApp (MVC). Keep in mind that WebApi and WebApp (MVC) now use the same framework unlike previous versions of the .NET framework. He also mentioned that an Angular (single-page application – SPA) was available though it required Node.js.

As a side note, Steve mentioned that the ASP.NET Core 2.0 team uses xUnitTest for their unit tests. While you can choose any test framework you like (I usually use the default UnitTest framework), it's a good idea to use the same framework the developers used.

Steve reviewed Razor pages, WebHostBuilder, startup configurations, Middleware, Dependency Injection, Tag Helpers, and much more. There's a lot to take in and I would not do you any justice going into more detail here, so check out his resources.

# All Requests are Asynchronous – Michael Perry

Michael explained how asynchronous messages across a network work through an entertaining example explaining how two generals on opposite sides of a castle decide when to attack by sending runners between camps with some unfortunate not to make it. To avoid a lot of handshakes and potential lost messages the requester changes its state and sends the request to the service which changes its state and then sends a response. It's key that you change your state before you send the message in case something fails on the state change the message will be correct even if it isn't received.

The [CAP theorem](https://en.wikipedia.org/wiki/CAP_theorem) in short states that you can only support two of three guarantees: consistency, availability, partition tolerance. Given that we are discussing communications between services over a network we have to support partition tolerance and we have to choose between consistency and availability.

The standard TCP/IP three-way handshake resembled the example with the generals. He reviewed how [Idempotent](https://en.wikipedia.org/wiki/Idempotence) functions can apply the same function multiple times and still get the same result and how the asynchronous messages should be treated the same. He also described the [AMQP architecture](https://www.amqp.org/product/architecture) at a high level. Finally, it led into his discussion of [Historical Modeling](http://historicalmodeling.com/) promoting a better way of communication across a network. This was an interesting session.

# Emerging Technologies and Thinking Different – Keynote – Ian PhilPot

The discussion revolved around the experiences of Ian's team ranging from monitoring pH levels of crops to monitoring the environment in a test disaster zone to facilitate emergency personnel in rescuing survivors. They were great stories that touched on the Internet of Things (IoT), serverless functions, and AI.

He emphasized the opportunities available to developers to innovate such as using virtual reality gear to provide training and specialized technical assistance across great distances saving the cost of travelling and wait times. This was another great motivational keynote to get everyone excited to learn.

# Rise of the Bot: Building Interactive Bots with Language Understanding – Brian Sherwin

Let's start it off by stating that Brian is an excellent speaker and I enjoy his style of presentation. As the former president and regular presenter at the [Central Ohio .NET Developers Group (CONDG)](http://condg.org/) I've had the pleasure of attending several of his sessions. He differs from other speakers by actually walking you through the steps to accomplish a task (of course with a backup PowerPoint presentation if something goes awry). Afterwards I usually feel that I have sufficient knowledge to reproduce the steps myself.

We were quickly led to the [QnA Maker (preview)](https://qnamaker.ai/) where we created a FAQ bot in minutes. I've already done this myself and found it pretty easy to do, though it did require me to grant some permission to my Azure account. The example consisted in navigating to <https://qnamaker.ai/> > Sign in with your Azure account > grant permissions > click Create new service > service name: TechElevatorBot; url(s): <https://www.techelevator.com/faqs> > Create. Yep, you're done.

So what did it do? It screen scraped/parsed html the questions and answers from the FAQs page and built a dictionary. You can review the Knowledge Base to see the questions and answers and make adjustments. Click Test and enter questions to see the results. If the question doesn't get the expected results, then you can choose the better answer or enter a new answer. The bot uses the Azure Language Understanding Intelligent Service (LUIS) under the Cognitive Services to interpret the question. I tried it against another FAQ page of dictionary terms and definitions, but it couldn't read it. I'm not sure if it was due to no question marks or the dt/dd elements, so there's definitely a pattern that must be followed for this to work. You can always resort to entering the data manually.

That's a lot already and if that wasn't great on its own, Brian even walked us through creating a bot in Azure. He went into a lot of detail and yes showed the steps to define intent, utterances, entities, and updating the utterances with entities (variables). When it's all said and done you can edit the code (I chose the C# template) online or download it. Even better, each Intent has an event defined with an attribute of [LuisIntent("{intent}")] allowing you to add logic to handle the request. I'm still working on it but it looks like you can check it into source code and setup Continuous Integration and Continuous Deployment. This was another of my favorite sessions!

# Why You Shouldn't Worry About AI Until You Should – Chris Slee

Chris started the first quarter of the session reviewing over a dozen AI related movies with the question in mind: Is the AI good, neutral, or evil. Some contested that if the AI kills a human regardless of motive, programming, or mission, then the AI was evil. This opinion was in the minority as most felt that the AI were good or generally neutral as they did not have intent to do evil and were only doing what they were programmed to do. I still hold out reservations for WALL-E after all he had a lot of replaceable parts in that transport vehicle, just saying.

Chris gave a high-level overview of AI and ML discussed in other session reviews above. He referenced multiple projects for review: [DeepMind](https://deepmind.com/), [IBM Watson](https://www.ibm.com/watson/index.html), [OpenAI](https://openai.com/), [Cognitive Services](https://azure.microsoft.com/en-us/services/cognitive-services/), [TensorFlow](https://www.tensorflow.org/), [OpenAI Gym](https://github.com/openai/gym), [Universe](https://github.com/openai/universe), [Azure Machine Learning](https://azure.microsoft.com/en-us/services/machine-learning-studio/), and [Microsoft Bot Framework](https://dev.botframework.com/).

Cognitive Services were broken down by Vision, Knowledge, Language, Speech, and Search. Review the [Cognitive Services](https://azure.microsoft.com/en-us/services/cognitive-services/) for details on each type. These links will give you a clearer high level overview.

<https://x.ai/> is a personal assistant who schedules meetings for you. Chris highly recommended [TED Talk: Can we build AI without losing control over it? – Sam Harris](https://www.ted.com/talks/sam_harris_can_we_build_ai_without_losing_control_over_it). I watched it and found it intriguing.

# Make Mobile Apps Great Again: NativeScript and Angular – Nick Branstein

Nick has posted his slideshow on gitub <https://github.com/NickBranstein/Presentations>. This was a good session on learning how to migrate a HTML and Angular site to a NativeScript and Angular app. The session title and description wer misleading as I thought this was going to be an intro on setting up NativeScript and Angular and writing your first app.

See the [NativeScript](https://www.nativescript.org/) site to get started installing Node.js, NativeScript CLI, and iOS and Android requirements. Note that you still have to use a Mac in order to compile for iOS. Android will compile on Windows or Mac.

Turns out most of the logic migrates one-to-one to corresponding files. The exception to the rule is the HTML UI code which needs converted to NativeScript UI. Div and span tags convert to layouts such as StackLayout, grid, wrap, absolute and more. Label, p, and h# convert to label. Button and input convert to button, textfield, datepicker, listpicker, etc. Lists can be represented with ScrollView.

While you don't use bootstrap classes, NativeScript has implemented most of the CSS classes already so they should convert. Events are slightly different such as (click) is (tap).

<https://play.nativescript.org/> allows you to quickly and easily test your NativeScript code. [The NativeScript Book – building mobile apps with skills you already have – by Mike and Nick Branstein](https://www.nativescript.org/get-the-nativescript-book) is a free eBook written by NativeScript experts.

# Azure Service Fabric: Live Large, Crash Never – Richard Broida

Richard from [Bennett Adelson](https://www.bennettadelson.com/) reviewed the benefits of using Service Fabric as an orchestrator of services across a managed cluster of machines. Service Fabric monitors and automatically repairs application hosts. It can be run in Azure, on premise, or other cloud environments.

Service Fabric provides wide choices for hosting, choice of programming model, deep tool integration with Visual Studio and Visual Studio Team Services, and is a mature and proven solution. In comparison, Serverless is consisting of logic apps, functions, and event grids are only hosted in Azure, PaaS only, limited programming models and tool integration, is getting more mature, and even without servers you still have to manage the app.

Service Fabric supports stateless and stateful services with [Reliable Services](https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-introduction). Stateless services are supported externally from the app such as in a database. Stateful services are supported internally such as queues. With stateful services if the service fails anything committed is kept and everything else is rolled back.

Richard opened VS2017 and demonstrated creating an on premise Service Fabric project under Azure Development. You can choose to create [stateless and stateful services](https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-services-quick-start). Tip: SF stores up to 10GB of cache in C:\SfDevCluster. You can't change the location so make sure you have enough space to support this. When you're not running the app on your dev machine, just delete the folder.

The rest of the session was reviewing the [.NET Service Fabric Voting sample app](https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-quickstart-dotnet). He recommended using DevOps tools to automate using PowerShell scripts and ARM templates instead of the standard Publish option for dev testing. Of special note, he was able to change a single value in one of the services and deployed. You could see each instance of the service (node) slowly implement the change as it received the update.

I may be misunderstanding how to use Service Fabric at this point, but it seems like you have to compile the services with Service Fabric for all the magic to work properly which seems limiting. I'll chalk this up to further research for now.

# Conclusion

DogFoodCon was a great experience. The [Quest Conference Center](http://www.quest-centers.com/) served most sessions very well with only a few getting crowded. The sessions were on par with what I experienced with StirTrek earlier in the year with the benefit of lasting two days if no movie.

I thank the presenters and orchestrators of this fine even and eagerly wait to do it all again next year. The above was a huge brain dump and consolidation of what I learned. As such, please correct or expand on any topic in the comments. Thanks!

We learned a lot about how to use artificial intelligence, machine learning, and bots along with other technologies, tips, and tricks. With this large download of knowledge we no longer fear the Robot Overlords today, though tomorrow is another day.