Stir Trek 2017

By Andrew Hinkle

Stir Trek is a great conference for introducing developers to the latest trends, standards, and especially for addressing pain points. The icing on the cake is to watch the currently trending Sci-Fi blockbuster Guardians of the Galaxy Vol. 2. Seriously, we come for the sessions… I am Groot!



# Stir Trek

I had the pure pleasure of attending 6 of 40 different sessions with breakfast and lunch provided along with tickets to see a blockbuster movie with soda and popcorn provided. Can't get much better, can it? Oh yes, it can! It's actually quite affordable at $99.

# Location, location, location

There was a little jump in price from previous engagements held at the Rave which was a standard sized movie theatre in Polaris OH. The cost increase is associated with the conference's continual growth which required the venue changed to The Schottenstein Center, Ohio State University, Columbus OH.

Right off the bat, we all missed the vastly superior movie theatre seats of the rave with the comfortable chairs, elbow room, and space. The seats packed us in as sardines and it must have been worse for those who weren't athletically inclined. I nearly tripped down the uneven depth of the arena steps multiple times which are very unfamiliar as this was probably only one of a few times I've been in any type of an arena. For the speakers, one accurately stated he felt like he was in an orchestra pit compared with the audience. I feared they were all going to get whiplash from looking up from their laptops to address the audience and then back to their laptop.

The sections were separated by black curtains on the left and right that rose well above our heads even in the back row. Behind the speakers was another black curtain that rose only to the height of middle row of seats. This curtain was used as a screen with the projectors on the other side. The displays were clean and crisp, but to the dismay of several presenters laser pointers had no effect. The screens doubled for privacy and sound reduction.

In regards to sound, the curtains were mildly effective as long as you were below the middle rows. Even then the acoustics still resulted in many distracting sounds from the other sessions for the speakers and audience, but at least in the lower level seats you could clearly hear the speaker. The upper level seats were a much poorer experience and if you were unlucky to be at the section entrance you might as well have left to speak with the vendors. I normally visit the vendors, pick up some goodies, and make a few connections. However, given the seating arrangements, I thrived to get to the sessions quickly so I could get a decent seat.

At this point you must be wondering why they changed the venue to have this poorer experience when the movie theatre was so much better for the sessions. Ah, you haven't been to the previous Stir Trek conferences then. We gave up the lap of luxury for the ability to actually walk between sessions without fear of drowning in the pool of people moving between sessions. You probably also weren't one of the poor unfortunate souls who had their cars towed away. Due to the utter confusion of attendees being refused parking in front of the theatre so "paying customers" could still park resulted in the conference goers parking at the local restaurants and stores that rightfully called foul as we REALLY weren't their "paying customers".

The parking availability compared to the previous experience was lush and vibrant. The buses from the arena to theatre were on time, prompt to pick up and drop off. There was confusion at the theatre regarding which line to pick up popcorn and drink which could have been easily remedied by putting the cups and bags of popcorn on the table they already had setup upon entry.

That's enough about the venue. Let's talk about what truly makes this conference great, the movie! Ok, fine. The sessions!

# Sessions

Go to <http://stirtrek.com> to get the schedule for all of the sessions. Stir Trek 2016 recorded the sessions and published them to the [Stir Trek Conference YouTube channel](https://www.youtube.com/channel/UCdflS38aeqXfuCTbM2I2S5Q) for your review. I'll review the 6 sessions of 2017 I attended this year.

## Understanding SQL Server Execution Plans by Drew Furgiuele

I enjoyed his presentation and left with knowledge learned that I will actually apply at work. First off, you have the distinct pleasure to watch his conference from 2016 even if you missed it this year. There is a minute in there when we were all told to move our cars before they were towed, again why we had a different venue this year.

[Exploring Microsoft SQL Server Query Execution Plans – Drew Furgiuele](https://www.youtube.com/watch?v=3LTcCOLFurc)

Drew clearly explained how query execution plans are created, associated cost of the execution is calculated, and how they are cached. You'll learn that Scans read every row and Seeks read a shorter list of rows. Predicates like Where can cause a Scan even though you would expect it to perform a Seek, because your Where is on a column that wasn't sorted so the Scan is more efficient. Merge Joins, Hash Match, and Nested Loops are also explained. Finally, he discusses parameter sniffing. Great info!

If you take nothing else from the lesson, Drew wants you to take away that SELECT \* and CURSORS are bad. This is an excellent session, so bookmark the YouTube video above as a reference whenever you need to improve the performance of your queries.

## Improv(e) your Testing by Damian Synadinos

<http://www.ineffable-solutions.com/current-offerings>

<https://www.slideshare.net/dsynadinos/iyt-codemash-2017>

Damian illustrates the relationships between comedic Improv (Improvisation). By following certain Improv rules you'll have the right mindset for approaching test scenarios. The "Yes, And" rule boils down to testing a behavior, record it, and then test around it. The "No Driving" rule enforces that you shouldn't allow your biases for an expected result deter you from recording your tests as your assumptions of what is correct may be wrong. Don't steer your tests to only return the desired behavior. There are more rules to adhere to, but this gives an idea.

While there are valid correlations the presentation left much to be desired. The workshop is supposed to be 1-4 hours as mentioned on his site, however, I would not recommend the 1 hour session without adjustments. The first 5 minutes was a justification and back story of why he is qualified to do Improv which may be ok for a workshop, but time is limited here and I prefer action to words. The next 15 minutes was a dictionary lesson on the terms Improv and Testing.

Incorporate terms into the presentation. Don't drag me kicking and screaming to a conclusion that words are related. This was as bad as the Domain Driven Design lecture I attended last year. Just get your point by demonstrating it. I wish presenters would stop this practice, just stop.

The audience participation was interesting and presented a couple laughs, but took time. In the end he barely touched on 3 of the 10 Improv rules we should have learned. Damian's workshop does have potential for a longer session that isn't rushed.

## Securing Your API Endpoints – A Practical Guide to API Authentication by Seth Petry-Johnson

<http://www.petry-johnson.com/blog/>

<https://www.youtube.com/watch?v=NHeoCocs60I>

I attended this session last year and was very impressed with Seth's presentation. So much, I'm mentioning it here even though I didn't attend this time. You'll learn about OAuth, API Keys, HMAC authentication and more. Bookmark and watch it.

## Essential JWT (Json Web Tokens) by Todd Nussbaum

<http://tnussbaum.github.io/index.html>

Todd did a great job introducing the audience to Json Web Tokens and possible usages. He described the following typical usage. The Client would send the username and password to an Authentication service that would create and return the JWT. The Client would then send the JWT to the service to perform an action and how that service would verify the contents against the signature, determine by the claims if the user is authorized, and if yes would then process the request.

Json Web Tokens have 3 parts: Header, Payload, and Signature. A Json Web Token is open to view, protected against manipulation. This means that each piece is Base64 encoded, so the Header and Payload can be decoded and read. JWT is meant for usage in stateless micro services/Service Oriented Architecture (SOA) environments.

The Header specifies the type and algorithm. The Payload stores the claims which basically tell you what the user may access. There are Registered, Public, and Private Claims which are all detailed in the IETF RFC7519 specs. Good luck with reading that, even the speaker only understands half of it. Even though the Header and Payload are open to Read, you can still encrypt claims, nest tokens, etc to your heart's content.

The Signature is the Header and Payload encrypted based on a secret key using HMACSHA256 encryption or another encryption method you choose and is then Base64 encoded. The Signature is strictly used to verify that the Header and Payload have not been manipulated from when it was created to when it was delivered.

## The Saboteur in your Retrospective by Arthur Doler

[The Saboteur in your Retrospectives: How Your Brain Works Against You (Stir Trek 2017)](http://bit.ly/2pb38Kb)

[The Saboteur in Your Retrospectives: The Handout](http://bit.ly/2o9kDxM)

I went into this session begrudgingly as another session had caught my eye. However, our team's Retrospectives have been less than spectacular and we felt there was room for improvement. I was pleasantly surprised at how well Arthur not only caught my attention; he kept it through the entire hour. If you happen to read this article Arthur you should definitely thank that user who requested through Twitter to add some hand drawn pictures. They were spot on!

I highly recommend everyone step through his slideshow if for no other reason to see Noodles. The slideshow contains enough detail that you should be able to follow along. The gist is that when we are presented with a question or issue we have instinctual/gut reactions aka Noodles that are totally biased by what has just happened and previous experiences. As an example, if you start off by framing the discussion with a negative reaction in your retrospective, then the rest of the retrospective may only be negative. Same if you went positive. Unfortunately, you really need to hear both sides.

Definitely checkout the "Then Handout" link above as it gives the short list of Do's and Don'ts. Some of the references may only make sense if you review the slideshow so keep that up on the other screen. It's great that he was able to identify issues during a Retrospective and provide direction on addressing them.

Arthur provided many personal experiences and examples for many of the common situations resulting in blame games, rock stars, nemesis, framing (anchoring), trying to control others, hindsight bias, information bias, etc. I found his presentation to be enlightening as we have all seen these biases to one extent or another. I wouldn't know of course, in hindsight I've always been a rock star in total control. :P

## Breaking Dependencies to Allow Unit Testing by Steve Smith

<http://ardalis.com/>

<https://github.com/stirtrek/2017MayStirTrek>

Watch [Real World Design Patterns by Steve Smith](https://www.youtube.com/watch?v=MpRQcCVZu14) from Stir Trek 2016. Steve is awesome, no seriously, watch his presentation. If you have PluralSight, watch [Refactoring Fundamentals by Steve Smith](https://www.pluralsight.com/courses/refactoring-fundamentals). If you don't have PluralSight, get a free trial and watch it, totally worth it.

There are 3 things developers are bad at. 1: Naming things. 2: Off by 1 errors. :D

Great jokes that never get old aside, naming things is hard. Steve was kind enough to give us the standard he follows.

Test Class Name: ClassNameMethodName

Test Method Name: DoesSomethingGivenSomething

It's much better than CustomerTests.cs with a method name Test1(). Usually, I go one step further with the Method Name by adding a Then Something to describe the expected result whether it is a value, response, or exception. I also usually add misspellings that my colleagues will correct me on later.

Test Method Name: DoesSomethingGivenSomethingThenSomething

He introduces the concept of Seams that represents areas of code where pieces can be decoupled. Testable code has many seems while legacy code has few if any other than the actual endpoint. To create Seams you must inject dependencies to reduce coupling so that all dependencies point towards infrastructure.

Thus high-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

If you follow the standard layer architecture, then it typically flows from Presentation layer to Business layer to Infrastructure layer. However, with dependency injection with a decoupled system your architecture layer should flow with Presentation layer and Infrastructure layer dependent on the Business layer.

Your code should have many Unit Tests, some Integration tests, and very few UI tests. UI tests depend on the Presentation layer. Integration tests depend on the Infrastructure layer. Unit Tests depend on the Business layer only.

I'm very passionate about learning how to improve my coding techniques and thoroughly believe that by learning to decouple my code to create unit tests properly improves the architecture of my projects as a whole. There is SO MUCH more to learn here, so finish this article real quick and start watching Steve's videos. Yes, they are that good.

## Fetch and Service Workers by Jared Farris

<https://jaredthenerd.com>

<https://jaredthenerd.com/2016/11/fetch-modern-async-server-calls/>

Jared demonstrated the future of API calls from Javascript using the Fetch API to make a generic Request and Response that we can use to GET/POST/PUT/DELETE/HEAD data to and from the server. The Fetch API replaces the XMLHttpRequest (XHR) with a much simpler implementation. Modern browsers Chrome/Firefox/Chrome/Safari/Edge already provide support, however, if you have to support IE 11 or older you may want to reconsider.

Service Workers are an answer to connectivity issues on the web. They are a middle ware service that can cache assets for offline use, manage network traffic, build retry mechanisms, and enable push notification. Service Workers are stored in the same folder as the content (domain) giving access to that folder and sub-folder, but not other domains. Service Workers are by domain not page. A Sync feature was also previewed though it is experimental at this time.

Debugging the Service Workers in Chrome is your best bet. When debugging in Chrome, the browser tends to frustrate Jared by forgetting about the Service Worker at times requiring him to close and reopen Chrome.

My take away is that if your client base is using the modern browsers and you are using Javascript to make API calls, then start learning the Fetch API and Service Workers. If not, then stare down your client-base with complete egocentric arrogance in your superiority until they cave into upgrading their systems to support modern browsers. That would work, right?

# Summary

All my hopes and dreams for the conference were met; Guardians of the Galaxy Vol. 2 was great! Oh, and Stir Trek was another great learning experience with so many great sessions. I hope they will post the sessions online again as they are great resources to bookmark. As most developers would say, to the operation team's dismay, the problem wasn't with the session it's the infrastructure that's to blame. Hmm, probably should review my notes on the Retrospective session again before I submit this. Na, I'm sure I'm right.