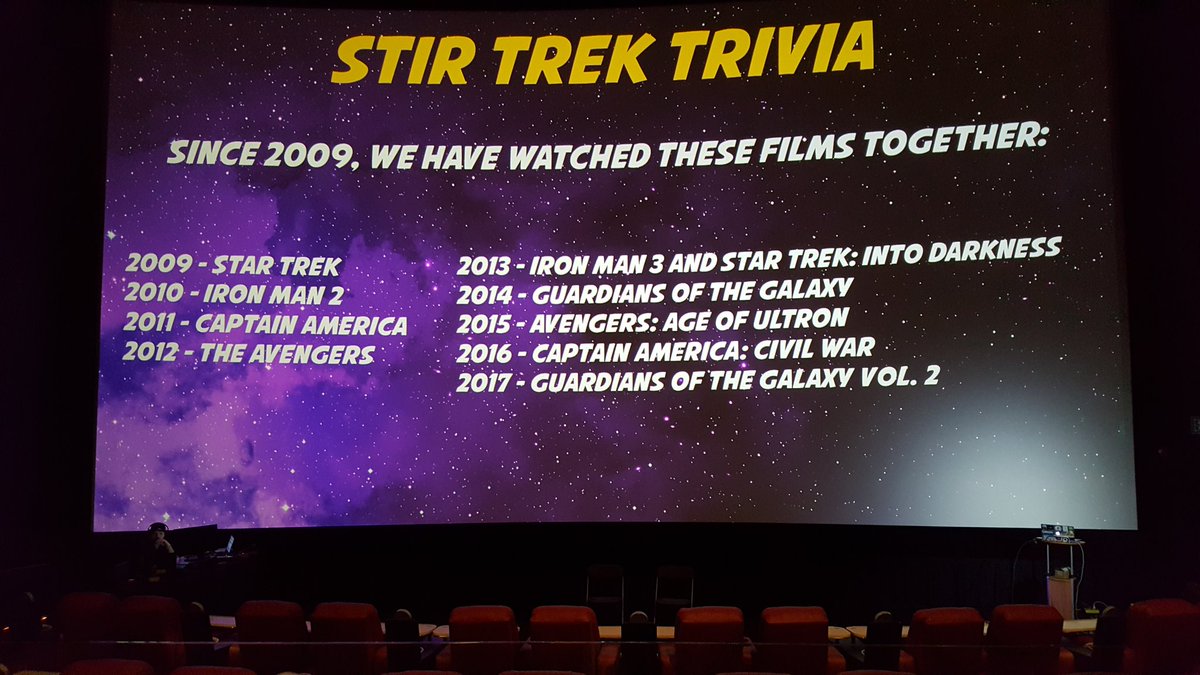
Stir Trek 2018

By Andrew Hinkle



Another great [Stir Trek](https://stirtrek.com/) convention has left my mind reeling and exhausted. Just the way I like it. Of course nothing beats the experience of being at the convention on site, but you can't be any multiple sessions at once unless you're Dr. Strange. Thankfully the coordinators of the conference have provided a treat by posting all of these great sessions on YouTube <https://www.youtube.com/c/stirtrek>. Give them a week or two to upload all the sessions, though I see a few up already.

# Location, location, location

If you read my [Stir Trek 2017](https://www.danylkoweb.com/Blog/stir-trek-2017-JL) post from last year or attended it yourself, you'll know that Stir Trek has been outgrowing its britches and struggling to find the correct venue. Stir Trek 2016 was at the Rave (smaller movie theatre), but with a record attendance parking was an issue. Yes, best to leave it at that. Stir Trek 2017 was at a convention center and as my previous article and the [Making of Stir Trek 2018](https://www.youtube.com/watch?v=BMyruwZQx-o&t=24s) video makes it very clear, that was a mistake.

This year, Stir Trek returned to its roots at the Easton AMC Movie Theatre and it was a great decision. While crowded, I never felt like a sardine like in the previous locations. Each speaker presented in one theatre and it was projected into adjoining theatres. Great concept and well implemented in most cases.

I personally did not experience any issues, mostly because I randomly chose the speaker's theatre 5 out of 6 times. I did hear a few complaints about technical difficulties for the first session from passersby's. The worst I experienced was a presenter having his mic occasionally get static from rustling across his shirt when he moved or not having the mic close enough until a coordinator assisted. I did find it funny when one of the presenters kept using a laser pointer to highlight a bullet point on the screen that no one in the other theatres could see. I give them a pass since this is a new format for most speakers.

There were plenty of vendors and recruiters with breathing room for all which was a welcome change. A quick shout out to all of the sponsors, for it's with their generosity that we may attend these conferences with reasonable rates. Rates did increase from $99 to $125. There's no doubt that this is a reasonable price in terms of tuition, gift box, food and refreshment, and of course the movie. The added benefit being that all the sessions will be available on YouTube means I'm not paying to attend 6 sessions, but all 48 sessions. I do caution that this year, unlike the previous two, the discussion on price is changing from no brainer to reasonable with trepidation that next year may price them out of going.

# Sessions

The number of sessions this year rose from 40 to 48. The recurring pattern for me is one time slot with three or more sessions that caught my interest, one time slot with one or none of interest, and one session that I attended fell short or I discovered too late to switch did not meet my expectations going in. This year continues the pattern. While I would love to rearrange the sessions to meet my own agendas I know all too well, I'd screw up someone else's. I don't envy the coordinators in deciding when each session is run. Thankfully, with each session eventually being posted in YouTube I'll get to see them all in my own good time.

## Boom! From Combat Engineer to Software Engineer by Steve Smith

Best session of the day! What a great start to the day with some [Metallica – The Day That Never Comes](https://www.youtube.com/watch?v=K6AJuRK2NE4)! Rev'd up on high octane music and caffeine, call me a fan already I'm pumped for the day! Yeah, that's a lot of exclamation points. I better stop before the key pops off my keyboard.

Steve detailed many relationships between his experiences as a Combat Engineer and Software Engineer such as mobility, counter mobility, and survivability. He described the military term [force multipliers](https://en.wikipedia.org/wiki/Force_multiplication) and how it applies to development as factors that dramatically increase the effectives of your software or team. The entire session revolved around this concept with real world examples from both perspectives. With plenty of illustrations and videos he had our complete attention.

Mobility involves removing obstacles such as removing barbed wires or mines. In software, this could be the removal of single points of failure. Example: Your software could use Amazon queues and if the services go down your software could rollover to use Azure queues until the primary service comes back up. He reinforced this flexibility with the unofficial motto followed by our military "[Semper Gumby](https://en.wikipedia.org/wiki/Semper_Gumby)" or "Always Flexible".

By far though, the greatest force multiplier he discussed was the tips on leadership. The military trains everyone to be leaders, willing and able to take control of a situation, because you never know when you'll be separated from your leaders and you'll be the highest ranked member of the team remaining.

Followership is a leadership quality. You should take orders as well as you give orders. Uniformity and standards directly relate to coding standards. If you don't apply a standard, you've just set a new standard. Lead by example. Your actions reflect on your organization even off duty. Take care of your team. Accept responsibility for team failures, share the credit for team successes, and call out positive individual efforts. Praise in public; punish in private.

His session was entertaining and worth attending and watching again once it's been posted. The discussion on leadership was by far the most valuable review and reinforcement of my own efforts as a team lead. Thank you!

## Hey, You Got Your TDD in my SQL DB! By Jeff McKenzie

I've been in the school of creating unit tests by mocking your dependencies for so long that the topic of this session caught me off guard. My experience to date for testing my queries and stored procedures (sproc) has been to add examples as comments at the top of the sproc with expected results. Of course in many cases the results will differ because the data changes, but you'd get enough of an idea of what was supposed to happen. If those words don't give you confidence that you could rerun the tests and get the same results each time, join the crowd.

The first quarter of the session was the typical bring everyone up to speed with basic terminology of what is Test Driven Design (TDD), why do we use it, so on so forth. I get it. As I've mentioned in previous reviews I prefer getting to the heart of the matter quickly. If you need to provide an intro or make a relationship between topics then do it quickly and move on.

The presenter setup a scenario for us similar to the "[Chrome Dev Tools: Raiding The Armory by Greg Malcolm](https://speakerdeck.com/gregmalcolm/chrome-dev-tools-raiding-the-armory) (2:15 session)". We were all now part of the "Apps 'n Stuff" team working for our client "Food and Stuff". It was a good setup and had some entertaining moments and visuals. I think I would have preferred the session to have started at this point, so another example or two of how to use the tool could have been demonstrated.

He proceeded to give a high level review of [tSQLt](http://tsqlt.org/), how to install, and use it. There are only a few files and they're fairly small. The test setup follows the standard "Arrange, Act, Assert (AAA)" pattern. The framework creates a transaction. Identify the tables that need tested and create tSQLt Fake versions of actual and expected tables. If there are any foreign key constraints that could cause a failure, you'll need to adjust the setup to use the real tables as the Fake tables drop all constraints. Insert data into the tables required for the test. Run the tests. Compare the results of the actual and expected tables to verify the results. The framework performs a rollback all changes.

All said and done, it's an interesting approach to testing queries and stored procedures I had not considered before. My team will definitely need to review this concept further.

## Lean Communication by Mandar Malunjkar

The presenter hit his points spot on with no waste epitomizing the definition of LEAN.

1. Show It
   1. Use pictures and diagrams to get your point across. Use a Whiteboard and draw it out. I find it very useful to help organize my thoughts as I'm explaining complex issues.
   2. Of course it can't get much clearer than these actual [IKEA instructions](https://biltapp.com/news/ikea-instructions-go-worst-best).
   3. While looking for the referenced image I ran across this video. Awesome! [Watch Ryan Reynolds Try to Build an IKEA Crib](https://www.youtube.com/watch?v=-wXjs6DXoeY&feature=youtu.be)
2. 5 emails rule (AKA Stop using Reply-All on emails sucking everyone's attention away from their work)
   1. Doesn't have to be 5 emails, it could be 3, 7, or whatever is best for your team.
   2. The goal is to cut off discussions via email or other media after roughly 5 replies. At this point, create a meeting and pull all necessary people into. Get it done instead of distracting everyone throughout the day.
3. [Mind Maps](https://en.wikipedia.org/wiki/Mind_map)
   1. Mind Maps are a way to organize information. Start with a topic. Define an idea related to the topic and draw a line between them. Define a sub-idea of the idea and draw a line between them. Many of us learn how to create these webs of thought in school for writing essays, stories, or just to organize our thoughts.
4. Squirrel (Jellyfish)
   1. Sometimes discussions veer away from the initial topic outside of scope. It could be as simple as discussing the different ways we could meet the deadline. Then someone brings up we wouldn't have the issue if we only had more environments. That's great; we could even setup the environments ourselves. Alright, that's great, but for the last half hour you've been discussing all the steps required to setup the environment, but it's 10PM and I haven't eaten since breakfast. Maybe we should get something to eat, that would surely help.
   2. Squirrel!!! As you can see, the discussion went sorely away from the original topic, and most certainly wasn't based on real life. Nope. Never happened. Hmm. So if you start to see the discussion go out of scope, use a "safe word" to bring the team back on topic. Jellyfish is fine, but I've always been called the squirrel. I mean the safe word is squirrel.
5. Bottom Line Upfront
   1. Give the headline (summary/highlight) of your message immediately like a newspaper article.
   2. However, do it from a positive manner.
      1. Starting off with "Extra, extra, we're not deploying tonight." This may come off as a little negative. Just saying.
      2. "We found bugs that Steve wrote. He's fixing them." This makes Steve look bad in front of the team and you look like an ass. Go watch the first session on leadership again, you need it.
      3. "While testing, we discovered some scenarios that don't meet our acceptance criteria. We're working them now." No blame, no shame, politically correct. You may still be in trouble, but you responded professionally.

I took the liberty of adding more examples and prolix, because I'm typically redundant, verbose, and wordy. As you might guess I'm really, really, REALLY working hard on making my communication LEAN, short, simple, and to the point. Upon review, I may need to take this session again.

## UX in an API by Kevin Mack

I misunderstood what this topic was supposed to be about. I wasn't alone in this. One of my co-workers left 20 minutes in. The person sitting next to me was as bewildered as me. Here's the part of the description I zeroed in on. "In this talk, we will discuss how to create a successful API that meets the demands of users and various outputs. The goal of this presentation is to help you write APIs and applications that will be performant for users." The goal was met in a very high level, surgically specific example that was discussed only in concept with no code.

The first half of the meeting was an introduction of himself, his multiple companies, presenting the argument that user experience developers are belong in the same team as developers. Coming from a background of only working with small businesses, I've been exposed to the standard "application/qa/data/infrastructure" silos and that's about it. I'm already in agreement being in the DevOps mentality working to bring collaboration between teams.

For short 45 minute presentations I feel the intro of yourself should be about a minute or two giving info on who you are, what you do, contact info, url to get content later, and move on. The longer intro may work better in longer sessions, but not in this format. One takeaway is that the presenter is very passionate about his work and genuinely seems like a great energetic person to work with. The discussion on the multiple businesses that he started illustrates his many successes and experiences. This gives you no doubt that he's an expert in his field and the topic at hand. Here's the rub, for each of these sessions all of the attendees of the conference already assume this and now we're halfway through the session and my co-worker already left.

The next quarter of the presentation discussed the complexities of implementing his product. Since the product revolved around liquor there were mounds of rules, regulations, and laws that must be adhered to. A great understanding of each condition was required to provide a great user experience which I think went to the heart of the topic. Gather requirements, understand what the user needs, and make it as seamless as possible for them.

It was at this time a few user experience flowcharts were opened and a single API contract in JSON was displayed. This is what I came for. How did you come to make decisions on how to present your data? API only, ties in with an existing web/mobile application. Cool. Versioning is important and there were lessons learned such as how to handle deprecating older versions. Use a token to identify each user. I'm excited now. Tell me more! Wait, wait, what? Times up and you can't go into the technical details now because it's outside of the scope of the presentation. Sigh, but that's why we came to this session.

## Giving Clarity to LINQ Queries by Extending Expressions by Ed Charbeneau

The topic revolved around taking advantage of the [Fluent Interface](https://en.wikipedia.org/wiki/Fluent_interface) common with [LINQ](https://en.wikipedia.org/wiki/Language_Integrated_Query) to change the low-level conditions to use static extension methods off [IQueryable](https://msdn.microsoft.com/en-us/library/system.linq.iqueryable(v=vs.110).aspx) to produce more legible LINQ statements.

Example:

postRepository.GetAll()  
 .Where(post => post.IsPublished  
 && post.PostedOn <= today);

Converts the complex condition to chain the expressions since multiple Where statements are treated as "And":

postRepository.GetAll()  
 .Where(post => post.IsPublished)  
 .Where(post => post.PostedOn <= today);

After applying the static extensions which you can learn more about in the session online once its available, the Where statements are refactored to the following.

postRepository.GetAll()  
 .ArePublished()  
 .PostedOnOrBefore()

This works fine until you try to perform an "Or". The presenter's solution is to take advantage of "[Expression Trees](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/expression-trees/index)" utilized under the covers by Entity Framework to build the SQL statements. At which point he could start adding .And and .Or statements with appropriate parenthesis. In the end it cleans up the code and the readability.

I'm not sure I'm quite on board with this approach, but I'm sure others will be happy to take advantage of it. I have no doubt it works. It feels a little dirty to overwrite the ORM, so my immediate thoughts run towards how to encapsulate this variation in case the ORM changes its functionality or I change ORMs these extensions would have to be updated to work the same.

Instead of calling the GetAll method straight off the repository and applying the filters from there, why not create another method in the postRepository or a unit of work that does this filter encapsulating all requests for the data in one place? The actual usage may fall in line with this separation of concerns, but it had me thinking.

The speaker had unit tests verifying the correct behavior which I'm assuming he did by faking the context. Since he's using static extension methods, you can't mock these methods and they're not in an interface for you to mock either. Faking the context would be necessary whenever you access this data making me want to encapsulate the calls as mentioned earlier.

I do find the approach interesting though and like the idea of using chain methods, but it wouldn't be as clean looking as what is shown here. Anyway, the session did its job to get me thinking.

## BFFs, Angular & .NET Core by Wasim Hanna

The presenter was kind enough to post all of the steps required to setup Angular and .NET Core to work well together with Azure Security & Authentication. <https://github.com/wfhanna1/AngularCore>

The presenter's goal was to build an app to manage a skills matrix that could be developed on multiple platforms (.NET Core) using Angular pages instead of Razor pages and had to use the preexisting Azure Security & Authentication. The first quarter of the presentation was an overview of ASP.NET Core, MVC, and Angular pages. Given the description of the session as a walkthrough of the architecture I was confused why it was included because at the very beginning of the session he said that the sessions had a prerequisite of "Working Knowledge of ASP.NET".

.NET Core starts very bare, so a package manager is used to install the required components. The session consisted of a lot of [NPM](https://en.wikipedia.org/wiki/Npm_(software)) calls to setup the solution in just the right order to avoid deleting files as part of the steps required that folders and files be merged into the solution after one or more steps. He included other tips to overcome obstacles that he ran into as well.

The Angular Pages were hosted on one port while the MVC was hosted on another port. He setup the MVC routing to redirect to the Angular Pages. At the end of the session in the MVC pages he wrapped the redirect with the following code to add the Azure Security & Authentication. He didn't have to add any further code to handle the authentication as all requests were served through the route.

@if (User.Identity.Authenticated) { … }

At the end I was still confused as to why the applications was sporting two different port numbers since the MVC only looked like it was a pass through to the Angular pages. Maybe it was the only way to form this hybrid that supported the authentication. In the end I think the session probably met its goals and I'm just not yet experienced enough in .NET Core to absorb everything.

# Conclusion

Stir Trek 2018 did not disappoint with the best location in the last three years. A great day of sessions followed up by a great movie with a fellowship of friends. I can't wait until all of the sessions are posted, so may catch up on the ones I missed out.

What sessions did you attend? What do you look forward to at these conferences? Prefer soft skills, code, something in between? What were the highlights to you?