Migrating ASP.NET Microservices to ASP.NET Core

By Example

Iris Classon

Apress®

Migrating ASP.NET Microservices to ASP.NET Core

By Example

Iris Classon

Apress®

Migrating ASP.NET Microservices to ASP.NET Core: By Example

Iris Classon
29 Gothenburg, Sweden

ISBN-13 (pbk): 978-1-4842-4326-8 ISBN-13 (electronic): 978-1-4842-4327-5
https://doi.org/10.1007/978-1-4842-4327-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484243268. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To the .NET community, my second home
and whose never-failing encouragement helped
keep my fire burning when I second-guessed myself.

Table of Contents

About the Author

Iris Classon is a force of nature. Her unique and engaging methods of teaching complex topics have garnered her considerable respect from the developer community and a great deal of media attention – Channel 9, Hanselminutes, Computer Sweden, and Developer Magazine – just to name a few. She is a Microsoft MVP and holds multiple certifications. Currently a freelance developer with her company, In Love With Code LTD, Iris can be found consulting for large enterprises and working on backend systems and operations for startups. She often speaks at conferences such as TechDays and NDC and at user groups. Her passion for teaching code extends to her tweets @IrisClasson, her popular blog, StackOverflow, MSDN, and a myriad of other social media sites.

About the Technical Reviewer

Sean Whitesell is a software developer in Tulsa, Oklahoma, and has been the president of the Tulsa .NET User Group since 2009. He is a frequent speaker at user groups and conferences. Sean has more than 18 years of experience in various aspects of software development ranging from Azure, Kubernetes, client-server, ASP.NET, Angular, embedded, and electronics development. His passions are in solving problems programmatically, coding craftsmanship, and teaching programming and martial arts.

Acknowledgments

I have to start by thanking my editors Jill Balzano and Joan Murray for their patience and support in writing this book. I had an exciting year with a move, travelling, belly growing, and childbirth, and Joan and Jill gave me flexibility when I needed it, focus and guidance when required, and most importantly, helped me stay on track so I could deliver this book.

My place of work, Konstrukt, has been vital for this book – not only by trusting me with publishing vital parts of our code but also by letting me share our journey.

But as cheesy as it might sound, the biggest thank you goes to my partner Emanuel Olsson. Dealing with a stressed programmer is one thing, but a pregnant and stressed programmer is a whole new ball game! Whenever I felt my motivation and energy drop, I could always turn to him for support.

Lastly, I want to thank my son, Loke Tiberiu, for keeping me company – kicking and rolling in my belly, burping and farting once out (and maybe a tiny bit of crying) – while writing this book. I couldn't have asked for better company!

Introduction

I remember when I was straight out of school how excited I was whenever I came across something new – a new language, framework, or tool – it was always fun and exciting to play around with it. After many years as a programmer, I still get excited, but I am also increasingly skeptical and hesitant to invest time in new things as I've seen so many come and go and not survive the test of time. This also goes for migrating existing systems. I used to want to rewrite everything, and I probably still do, but I've also seen how expensive this can get without necessarily yielding a better code base or system.

When .NET Core was first announced, I was enthusiastic but skeptical – even a bit cynical. I had jumped on the portable class library train early on and had my share of fun with that, and therefore the promise of .NET Core wasn't something I was going to believe until I could see it delivered. ASP.NET Core was the first framework to make use of .NET Core, and although I could see many benefits early on (often a result of getting to rewrite something that has collected technical debt over time), I wanted to wait and see where this would go. As the adoption rate increased, along with the contribution rate to this open source framework, we started discussing at work if ASP.NET Core could be something for us. The problems I came across while trying to answer that question were that information on ASP.NET Core was lacking, was referring to older versions, was scattered, inconsistent, and most importantly lacked real-world examples of doing large-scale migrations. If you, like me, are a natural skeptic, I can imagine that you would lean more toward a no in terms of migrating, even more so if you get lost in the information and opinion jungle, we spend most our time in. My aim with this book is to foremost show a real-world example, with actual code from our system, of a migration from ASP.NET to ASP.NET Core neatly organized in easy-to-follow steps. While the book won't be an authoritative guide on ASP.NET Core (there are plenty of excellent books that cover ASP.NET Core), it will cover what you need to know to make a decision on whether or not to migrate. It will cover your migration options, how to do an analysis, and prepare, migrate, and maintain your web services. The book also has an extensive list of resources and tools that can come in handy, as well as plenty of examples of both problems and solutions that you might come across. This book has all the information that I was struggling to find, and my hope is that this book will answer the majority of your questions and let you focus on building awesome software be that with ASP.NET Core or not.

A Bit of Background Material

When I first wrote the outline for this book, I didn't plan on spending too much time explaining ASP.NET Core. I would assume a certain level of understanding from you, the reader, and we'd hit the ground running. But, often as developers, we are pressed for time and decisions are made when we start implementing, with good intentions to at some point take the time to dive deep in a new technology, but as we all can attest, sometimes we never get around to it. Therefore, I want to take a little of your time in the beginning of this book to share my insights and cover some of the more valuable aspects of ASP.NET Core that you might not be aware of. Hopefully, this will allow us to create a solid base for the migration that lies ahead of us in this book.

Besides understanding .NET Core, .NET Standard, as well as ASP.NET Core, I also want you to have an insight into the history behind the Microsoft web development stack and how we ended up with ASP.NET Core. And an important part of that is covering the difference between ASP.

NET Web Forms, ASP.NET Web API, and ASP.NET Core. Not to mention the difference between the two project types: ASP.NET Core (.NET Core) and ASP.NET Core (.NET Framework).

Web Development Stack Timeline

Let's see if I can condense the history of ASP.NET Core and how we got here. If you have been around for a while, you know it's a long story that can be tracked back to when Classic ASP (Active Server Pages) was a thing, and maybe even further back to one of Microsoft's first inter-process communication methods, DDE (Dynamic Data Exchange), in the late 1980s. Classic ASP was a revolutionizing way of dynamically rendering server-side pages with baked-in logic, and later evolved to ASP 1.0, followed by ASP.NET and Web Forms in 2001. In 2009, ASP.NET MVC was released as a (much-needed) alternative to Web Forms. Simultaneously, communication methods evolved from DDE to DCOM, to .NET Remoting, to ASP Web Services in early 2000. In 2006, WCF (Windows Communication Foundation) was released, tying together the different options and supporting a variety of communication standards. WCF was intended to be a unified programming model for building service-oriented applications that had explicit support for service-oriented development. However, WCF had some limitations in regard to REST (Representational State Transfer) support. The WCF Web API project was intended to fill the gaps and eventually evolved and became what we today know as Web API. Web API was quickly embraced, partly because it tied in nicely with the ASP.NET MVC style of programming. Not so surprisingly, when ASP.NET Core was written, as a rewrite of ASP.NET, the two were combined (Web API and MVC). But how exactly do .NET Core, .NET Standard, and ASP.NET Core fit in?

Core and More

As I see it, there were two major driving forces that led to what we today know as .NET Core. Firstly, there was an increasing need for cross-platform compatibility to stay relevant, and secondly, we had gotten to a place where we had too many subsets of the .NET Framework, which was causing all sorts of problems as the subsets were created and maintained by different teams. 2014 was the year many exciting announcements were made by Microsoft. ASP.NET vNext was announced in April, and .NET Core in November the same year. .NET Core was a fork and open source rewrite of .NET, and likewise ASP.NET vNext was a complete rewrite of ASP.NET. It was later known as ASP.NET 5 (and by some as project K), until Microsoft realized the name was confusing and renamed it to ASP.NET Core.

In 2016, the .NET Standard was introduced to us, as a way to bring everything together. .NET Standard is a specification; it defines a set of available .NET APIs. You can think of it as interfaces and the frameworks that support the (or version of) .NET Standard as implementations. If you ever had the pleasure of working with Portable Class Libraries (PCL) and the framework union model they used, you'll be pleased to hear that .NET Standard will replace PCLs.

To sum up, ASP.NET Core is the next step in the web development evolution, an evolution that goes back more than 30 years. During that time a lot has happened; tools and frameworks have been developed and matured. The goal has always been to make it easier for us as developers to create performant, flexible, yet sturdy services that allow freedom in terms of how we communicate over the network and serve our content. ASP.NET Core puts together some of the last missing pieces in terms of combining the power of other frameworks while giving us the flexibility to choose the operating system best suited for our service and optimal performance all while encouraging best practices. You might not decide to migrate after reading this book or doing the suggested analysis, but the information will be valuable nonetheless as ASP.NET Core is the future of web development with the Microsoft stack.

CHAPTER 1

The SaaS System in Question

I never pictured myself working at a startup. For as long as I've been working as a developer, it has mostly been consulting gigs, and to me a startup meant unreasonable working hours, low pay, a bunch of hipsters arguing over what toys and games to stock in the conference room, plus little to no stability. But, never say never, especially in the tech world.

One evening, I joined a friend of mine for a late-night coding session with his business partner Johan (also a developer). They were battling performance problems and bugs into the wee hours of the morning.
One evening turned into several weekends in a row, complete with the pick and mix candy (a Swedish specialty), more caffeine than anyone should consume, and raucous music pouring out from the speakers.
We coded tirelessly in that office overlooking the city center, and I ended up offering to help profile the services they were experiencing problems with. The services were ASP.NET services with significant performance and memory use problems. At that time, the answer to the problem was to create a local, on-prem solution for each customer. Each customer had a local installation, not a big deal, as there were only two customers. But, understanding that their client list was growing, I think we all recognized the value of taking the system to the cloud to make a multitenant solution. The partners chose Azure, an excellent choice I thought, as I had a decent amount of experience with Azure and had been impressed. Then the

I. Classon, *Migrating ASP.NET Microservices to ASP.NET Core*,
https://doi.org/10.1007/978-1-4842-4327-5_1