ASP.NET Core 2.1 Fundamentals

Scott Sauber

Audience

- Existing ASP.NET developers
- People interested in the new ASP.NET Core stack
- People using ASP.NET Core 1 or 2

Agenda

- Split into 2 parts
 - Intro to ASP.NET Core in general
 - What's new in ASP.NET Core 2.1
- Demos all throughout
- Questions any time

Who am 1?

- Software Developer, working primarily with web and ASP.NET
- Worked with the .NET since 2009
- Avid learner
- Following ASP.NET Core since the early days
- Blog primarily on ASP.NET Core on <u>scottsauber.com</u>

Timeline

- May 2014 ASP.NET vNext announced
- February 2015 ASP.NET vNext named to ASP.NET 5
- January 2016 ASP.NET 5 renamed to ASP.NET Core 1.0
- June 27, 2016 ASP.NET Core 1.0
- August 14, 2017 ASP.NET Core 2.0
- April 7, 2017 ASP.NET Core 2.0.7
- April 11, 2017 ASP.NET Core 2.1 Preview 2
- Guessing ASP.NET Core 2.1 will drop Julyish 2018

What is ASP.NET Core?

- Ground up rewrite
- Modular, pay for play via NuGet packages
 - Improved performance
 - Downside explicit about wiring up what you need
 - Templates help solve this
 - Microsoft.AspNetCore.All/.App
- Open source
- Cross platform (Windows, Mac, various flavors of Linux, containers)
- It's just a console app
- Runs on .NET Framework and .NET Core

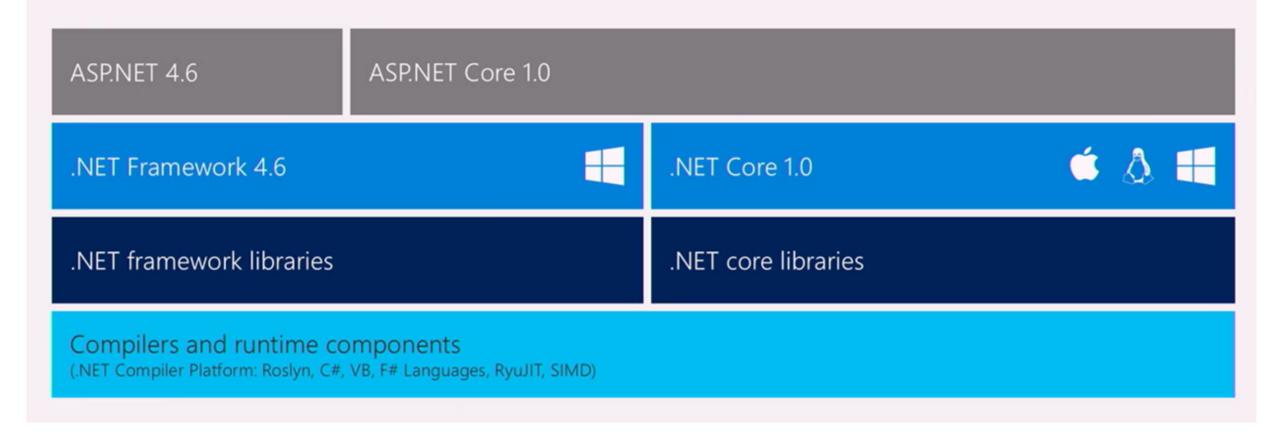
What is .NET Core?

- Cross platform .NET
- Trimmed down version of .NET Framework to get it to run crossplatform
- How to see if your code can run on .NET Core
 - .NET Portability Analyzer
 - Windows Compatibility Pack
 - DirectoryServices, Drawing, Configuration, EventLog, and more
- ASP.NET Core runs on top of .NET Core (and .NET Framework)
- Allows you to run diff versions side-by-side on the same server
- .NET Core 2.0 added a ton of API's from 1.0

NUMBER OF APIS

Version	#APIs	Growth %
1.0	7,949	
1.1	10,239	+29%
1.2	10,285	+0%
1.3	13,122	+28%
1.4	13,140	+0%
1.5	13,355	+2%
1.6	13,501	+1%
2.0	32,638	+142%

ASP.NET 4.6 and ASP.NET Core



Source: Microsoft

.NET Core LTS vs. Current

- Which one to choose
- LTS
 - Don't care about new features
 - Don't want to upgrade right away
 - Support for 3 years after release or 1 year after subsequent LTS release, whichever is shorter
 - Latest LTS version is .NET Core 1.1.8
- Current
 - Want new features
 - Willing to upgrade frequently
 - Support for 1 year after release or 3 months after subsequent Current release, whichever is shorter.
 - Latest Current version is .NET Core 2.0.7
- New major/minor releases every 6 months
- New patch releases every 1-2 months

New .csproj project system

- "SDK style"
- Still uses msbuild
- Manage NuGet packages in csproj
 - No more packages.config
 - No more hint paths
- Slimmed down csproj
 - Hello World down from ~300+ lines to <10 lines
- In the folder, in the project by default
 - Can manually exclude files
- Live Edit csproj without needing to unload and reload
- Only works in ASP.NET Core/.NET Core projects today

Old vs New .csproj side-by-side

MVC 5 Template



ASP.NET Core Template

9 lines (really 8 lines without blank lines)

321 lines

dotnet CLI

- What is a CLI?
- Restore, Build, Run, Publish, New project all from the command line
 - dotnet restore
 - dotnet run
- Installed with the .NET Core SDK (download at http://dot.net)
- Cross platform (Windows, Mac, Linux) with any editor (VS, Code, Sublime, vim, emacs, Notepad if you like pain, etc.)
- Works for any .NET Core app, not just ASP.NET Core

dotnet SDK

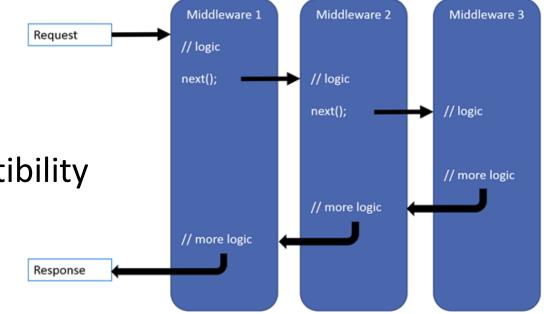
- Core libraries and msbuild tasks for what everything runs on top of (including VS, CLI, etc.)
- Also contains the .NET Core Runtimes (Current and LTS), all the Microsoft NuGet packages, and the CLI
 - Offline restores
- Different versions can run side-by-side
- Just a folder, easy to uninstall
 - C:\Program Files\dotnet\sdk
- One install to get you all the things

Let's get back to ASP.NET Core

- Important to understand .NET Core
- First choice when starting with ASP.NET Core.
 - .NET Core or .NET Framework
- Core name gets thrown around. .NET Core and ASP.NET Core are different.

ASP.NET Core – what's different than ASP.NET 4

- ASP.NET Core is simply a console app.
- Web Forms is gone
- WCF server is gone
- Client side package management
- web.config is only around for IIS compatibility
- Global.asax is gone
- Startup.cs
- Middleware all the things, even MVC
 - Order matters



Nice New Features over ASP.NET 4

- Save and reload, no more building
- Dependency Injection built-in
- TagHelpers > Html Helpers
 - HtmlHelper: @Html.ActionLink("Home", "Index", "Home", new { @class = "btn btn-primary" })
 - TagHelper: Home
- Environments are a first class citizen via IHostingEnvironment

MVC – Where's my cheese?

- MVC and Web API Controllers have been unified
 - No more ApiController, just inherit from Controller
- Child actions gone in favor of View Components
- Views/_ViewImports.cshtml is your new /Views/web.config
 - Instead of <add namespace="MyNamespace.Something"/>
 - Just @using MyNamespace.Something
- Static files now served by folder called wwwroot
 - I treat wwwroot as my "bin" directory. Source files live elsewhere and bundler puts them in wwwroot

MVC – What's the same?

- ASP.NET MVC Concepts are the same
 - Still have Controllers
 - Controllers still have Actions
 - Still have Views
 - Still have partial views
 - Routing
 - ModelState
- HTML Helpers still exist
 - But you should use Tag Helpers

Kestrel

- Brand new web server
- Built on libuv in 2.0
 - Built on System.Net.Sockets in 2.1
- Cross platform

Kestrel – On The Edge

- With 2.0, can now put it "on the edge" and be supported
- But they still recommend you still use a reverse proxy
 - IIS on Windows
 - nginx or Apache on Linux
- Why?
 - IIS took 7 years to harden. Kestrel is ~3 years old
- Barry Dorrans talk on Kestrel's Security (4:55 to 6:35)

So ... Kestrel on the Edge

Is Kestrel Edge Ready?

\(\(\varphi\)\)

We still recommend putting Kestrel behind a proxy
However 2.0 RTM will no longer be unsupported

Kestrel with Reverse Proxy



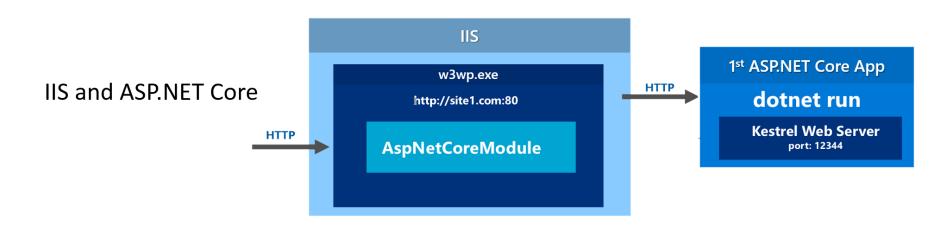


Image: Rick Strahl

Tech Empower Benchmarks

- What are the TechEmpower Benchmarks?
- https://www.techempower.com/benchmarks/
- ASP.NET Core 2.0 2.2M requests per second
 - Ruby on Rails: 29K
 - PHP: 39.5K
 - Django: 159.9K
 - Node: 525K
- This is a plain text this is essentially hello world
- Power of Open Source
 - #2 contributor to Kestrel, Ben Adams, is a non-Microsoft Employee
- This is where modularity is crucial

Let's talk about 2.0

- RTM'd on August 14
- A move towards simplicity
 - More defaults, less verbose
- Razor Pages
- New, Customizable Templating Engine
- TagHelperComponent
 - Inject in something (like JS/CSS/etc.) to beginning or end of Head or Body
- Precompilation of Views happens on publish by default
- Authorization got an overhaul

Program.cs is Simplified

ASP.NET Core 1:

ASP.NET Core 2:

```
public class Program
{
    Oreferences | O exceptions

    public static void Main(string[] args)
    {
        BuildWebHost(args).Run();
    }

    1reference | O exceptions
    public static IWebHost BuildWebHost(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
        .UseStartup<Startup>()
        .Build();
}
```

What CreateDefaultBuilder does

Microsoft.AspNetCore.All Simplifies Dev

- Metapackage
- Contains ALL the packages Microsoft ships (AspNetCore, EF, etc.)
- Simplifies upgrading to latest
- Simplifies in "oh I need another NuGet package now"
- Publish will "trim" out your packages you don't need
 - Utilizes the Runtime Store
 - More on this later
 - Publish will also pre-compile views by default
 - Improve startup time of views
- .NET Core 2 feature only (not full framework)

Microsoft.AspNetCore.All Simplifies Versions

- Single version instead of multiple
 - Why was it multiple in 1.0?
 - Didn't rev the package if the package didn't change.

ASP.NET Core 1:

<Project Sdk="Microsoft.NET.Sdk.Web"> <PropertyGroup> <TargetFramework>netcoreapp1.0</TargetFramework> <PackageTargetFallback>\$(PackageTargetFallback);portable-net45+win8+wp8+wpa81;</PackageTargetFallback> </PropertyGroup> <ItemGroup> <PackageReference Include="Microsoft.AspNetCore" Version="1.0.5"</pre> <PackageReference Include="Microsoft.AspNetCore.Mvc" Version= 1.0.4</pre> <PackageReference Include="Microsoft.AspNetCore.StaticFiles" Version=</pre> <PackageReference Include="Microsoft.Extensions.Logging.Debug" Version "1.0.2" /> <PackageReference Include="Microsoft.VisualStudio.Web.BrowserLink" Version="1.0.1" /> </ItemGroup> <ItemGroup> <DotNetCliToolReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Tools" Version="1.0.1" /> </ItemGroup> </Project>

ASP.NET Core 2:

Microsoft.AspNetCore.App

- Microsoft.AspNetCore.All will be deprecated with 2.1
- Replaced by Microsoft.AspNetCore.App
- Identical to .All other than removes packages not owned by ASP.NET or .NET teams
 - Microsoft.Extensions.Caching.Redis
 - Uses StackExchange.Redis owned by StackExchange team
 - Microsoft.AspNetCore.ApplicationInsights.HostingStartup
 - Owned by Applnsights team

Runtime Store

- Take assets from NuGet package and put them in a global location on the machine
- The GAC is back baby!
 - The NuGAC
- Optimizes the packages by CrossGen'ing that pre-JIT's the assemblies
 - Improves startup time
 - 1.x 3s
 - 2.x 1s
- By default, all Microsoft packages (Microsoft.AspNetCore.All/.App) are included in Runtime Store via .NET SDK install
 - Improves the Publish Size of App
 - 1.x 16MB MVC Hello World
 - 2.0 4MB MVC Hello World
- .NET Core 2 feature only (not full framework)

What's wrong with Controllers Today?

- Can easily get bloated with lots of actions and logic if you're not careful
- Inject in all dependencies for all actions, even though rarely does any action use all dependencies
- Can be hard to navigate
- Look at AccountController.cs in the templates
 - 500ish lines of gobbly gook

Introducing - Razor Pages

- Page focused approach over Controller focused approach
- I like this concept a lot.
 - Focusing on a single page, GET and POST
 - Dependencies just inject what that page needs
 - Rarely working on multiple actions/pages at once when working on a bug or feature. Almost always working on a single page.

Deconstructing Razor Pages

- View Page
 - Register.cshtml
 - @page at the top
 - Minor difference from MVC
- PageModel
 - Register.cshtml.cs
 - Code behind is back!
 - Inherits from PageModel
 - Think of it like a mini Controller, but it also includes your ViewModel
- That's it!
- Nothing special to wire up in Startup.cs. .AddMvc adds RazorPages
- Because it's just MVC under the hood.

Razor Pages Differences from MVC

- Controllers are gone
- Controller logic of GET, POST, etc. is now in a code behind file with Page Handlers like On<HttpVerb>Async such as OnGetAsync or OnPostAsync
 - Register.cshtml.cs
 - Async optional, could be OnGet or OnPost
- Default Pages folder is /Pages instead of /Views
 - Configurable via .AddRazorPagesOptions() off of AddMvc
- Top of view define @page
- Model binding is opt-in via [BindProperty]
- Improves folder structure

Razor Pages – "Replacing" HTML MVC (not API's)

- Razor Pages is now the default ASP.NET Core Web App template
 - MVC still available via an ASP.NET Core Web Application MVC template
- "We believe Razor Pages is a vastly superior way of doing server-side HTML generation." (than MVC) – Damian Edwards
- Can use MVC and Razor Pages side-by-side
- Razor Pages is just an extension to MVC. Uses same things MVC does
 - Partials
 - Model binding
 - Still a _ViewStart and _ViewImports
 - Because it's just MVC under the hood

Demo

- File => New Project => Razor Pages and let's have a look around
- Let's convert an MVC action to a Razor Page
 - Default Folders
 - Page vs Views
 - Code behind
 - Inherit from PageModel
 - OnGetAsync and OnPostAsync (and On<HttpVerb>Async)
 - As well as Sync versions, just drop the Async
 - Routing

Other Razor Pages benefits

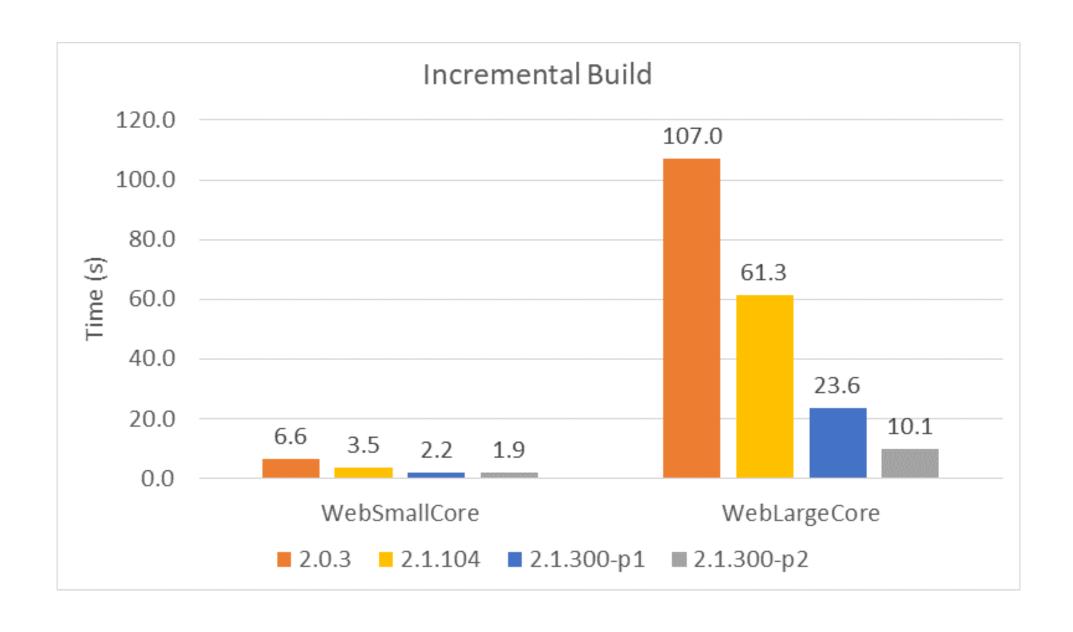
- Anti-Forgery Validation happens automagically in Razor Pages
 - No more [ValidateAntiForgeryToken] attributes
 - <u>Sidenote: if using ASP.NET Core MVC register the</u> <u>AutoValidateAntiForgeryToken attribute to your global filters</u>

My opinion on Razor Pages

- Love the concept of Page-focused vs. Controller focused
- It's still a bit early
- Good news here is most logic remains in tact between MVC =>
 Razor Pages, with just the few caveats I've shown
- I've used it on an Insurance system with over 50 pages and it's held up

Let's talk 2.1

- 2.1 currently Preview 2
- RC with go-live license coming within a month
- Full release coming in ~2 months I would guess



HTTPS Improvements

- HTTPS is default in 2.1 templates
- Easier to set up locally without IIS Express
- HSTS middleware in the box
- Cleaner to require HTTPS in 2.1

2.0 and below

```
var options = new RewriteOptions()
    .AddRedirectToHttps();
app.UseRewriter(options);
```

2.1

```
app.UseHttpsRedirection();
```

HTTP Client Factory

- Provides centralized naming and configuring of HttpClient
- Set default headers, base URL's, etc.
- Allows you to set up automatic retry, circuit breaker, perf monitoring, etc. globally

Share UI

- Ability to share Razor Pages (Views + Page Model) or MVC in a class library
- Example: Put Razor Pages in /Pages in class library, then just navigate to that route
- Override in your Web UI project by placing it in the same path
- Identity UI
 - Override via Right-Click => Add New Scaffolded => Identity, check what you need.
- Another use case: HTML Emails generated via Razor, consumed by another class library

SignalR Core

- Real-time communication (via WebSockets or other transport).
- Client => Server or Server => Client
- Most of the concepts are the same
 - Still have: Hubs, Clients, Groups
- No jQuery dependency
- JavaScript and TypeScript clients
- Less opinionated
 - Could be JSON or MessagePack or ...?
- Less features in the box
 - No more automatic replay
 - Old SignalR kept around 1K messages sent to a client, could be memory intensive

GDPR

- Some niceties to get you started down path of GDPR compliance
- Cookie consent middleware
- Download personal data

Global Tools

- Install global tools on your machine
- Replaces <DotNetCliToolReference>
- Like npm install –g <name>
 - dotnet tool install –g <name>
- Following installed with SDK on your machine
 - watch
 - ef
 - user-secrets
 - dev-certs

[ApiController]

- Opinionated way of setting up an ApiController
- Automatically send back 400 with ModelState when Validation doesn't pass
- No longer need [FromBody], [FromRoute] or [FromQuery] in most scenarios
- Requires attribute routing
- ActionResult<T>
 - OpenAPI/Swagger

Other things

- Improvements for functional testing
- Roslyn Analyzers
- Precompile Razor Views on build
 - Errors renaming a prop
 - Using that's not there

What's coming in 2.2? (subject to change)

- Templates
 - Move to Bootstrap 4.x (intelligent Scaffolding detecting Bootstrap 4 vs 3)
- LibMan
- Bundling & Minification middleware, build, and CLI tool
- Rename properties in Views... no more R#
- API Controller Conventions
- OpenAPI (aka Swagger) support
- API Client generation for C# and TypeScript
- Health Checks
- Distributed Configuration
- STS + API Auth
- HTTP/2 in Kestrel
- Performance of Kestrel
- Hoping for 2018

How to stay on the bleeding edge/latest

- Watch the ASP.NET Community Standup almost every Tuesday
 - http://live.asp.net for details
- Go to GitHub and select Watch on the Announcements repo
 - Only issues created are for announcing breaking changes
 - https://github.com/aspnet/announcements/issues
- If this stresses you out... don't do it. Just wait until things RTM and read the release notes.

How do I get started?

- Go to http://dot.net
- Click on Download
- Follow instructions for your OS of choice

So should I switch to ASP.NET Core today?

- "It Depends"
- You should definitely be evaluating it
- But ASP.NET 4.x is still going to be supported for a long time

Resources

- Look in the /samples on GitHub
 - Example: https://github.com/aspnet/Docs/tree/master/aspnetcore/fundamentals/configuration/sample/src
- How to get ASP.NET Core
 - http://www.dot.net
- ASP.NET Community Standup
 - http://live.asp.net
- ASP.NET Monsters
 - https://channel9.msdn.com/Series/aspnetmonsters
- ASP.NET Core Documentation
 - https://docs.asp.net/
- ASP.NET Core Source
 - https://github.com/aspnet
- ASP.NET Core roadmap
 - https://github.com/aspnet/Home/wiki/Roadmap
- .NET Core roadmap
 - https://github.com/dotnet/core/blob/master/roadmap.md

Questions?

 Feel free to reach out on Twitter (@scottsauber) if you think of a question later

• Slides posted on my blog (scottsauber.com) and I'll tweet them out

Thanks for coming!

Blazor

- What is WebAssembly?
- Mono can compile to WebAssembly and then run your C# code in the browser
- Blazor = Experimental .NET SPA Framework that runs client-side
 - Component-Based, Routing, Validation, DI, SSR, .NET debugging in IDE + Browser, and more
- AJAX with HTTP Client, not jQuery, fetch, etc.
- Validation problem today
- .NET Standard 2