ASP.NET Core 2.0 Fundamentals

Scott Sauber

Audience

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

Who am I?

- 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 scottsauber.com

Agenda

- Split into 2 halves
  - Intro to ASP.NET Core in general
  - What's new in ASP.NET Core 2.0
- Demos all throughout
- Questions any time

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
- Open source
- Cross platform (Windows, Mac, various flavors of Linux, containers)
- It’s just a console app
- Runs on .NET Framework and .NET Core

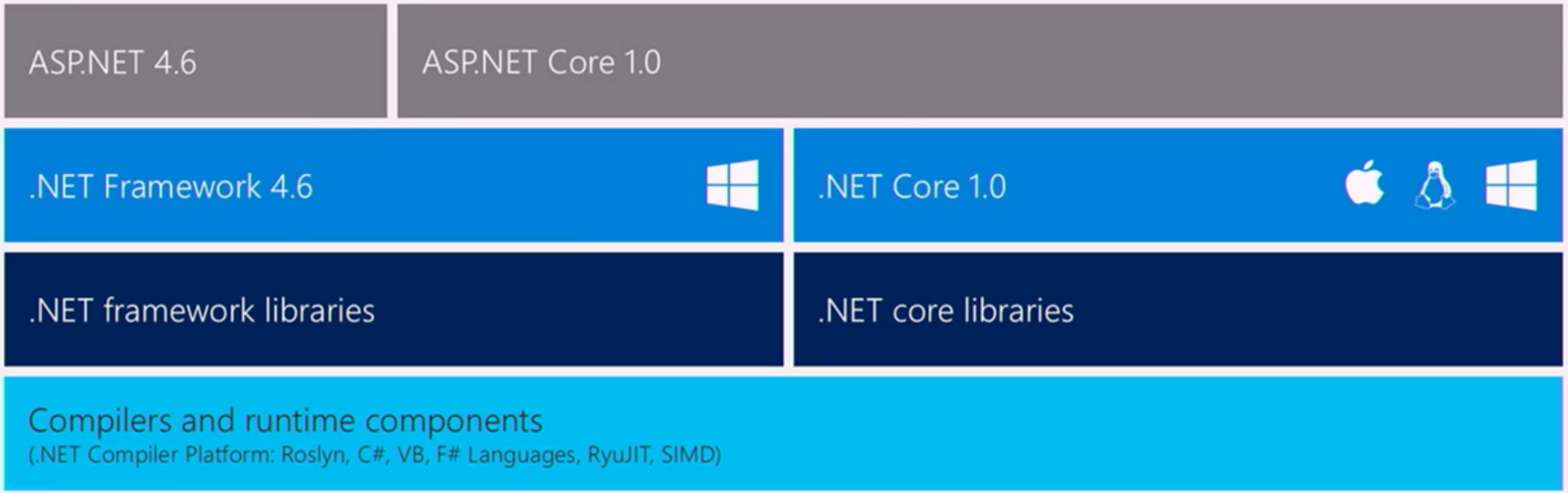
What is .NET Core?

- Cross platform .NET
- Trimmed down version of .NET Framework to get it to run cross-platform
- How to see if your code can run on .NET Core
  - .NET Portability Analyzer
- ASP.NET Core runs on top of .NET Core (and .NET Framework)
- .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 1.1.4
- 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 2.0.0
- New major/minor releases every 6 months
- New patch releases every 2-3 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
  - New csproj coming to older .NET projects later “this year” per Scott Hunter at DevIntersection

Old vs New .csproj side-by-side

MVC 5 Template

321 lines

ASP.NET Core Template

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.All" Version="2.0.0" />
  </ItemGroup>
</Project>
```

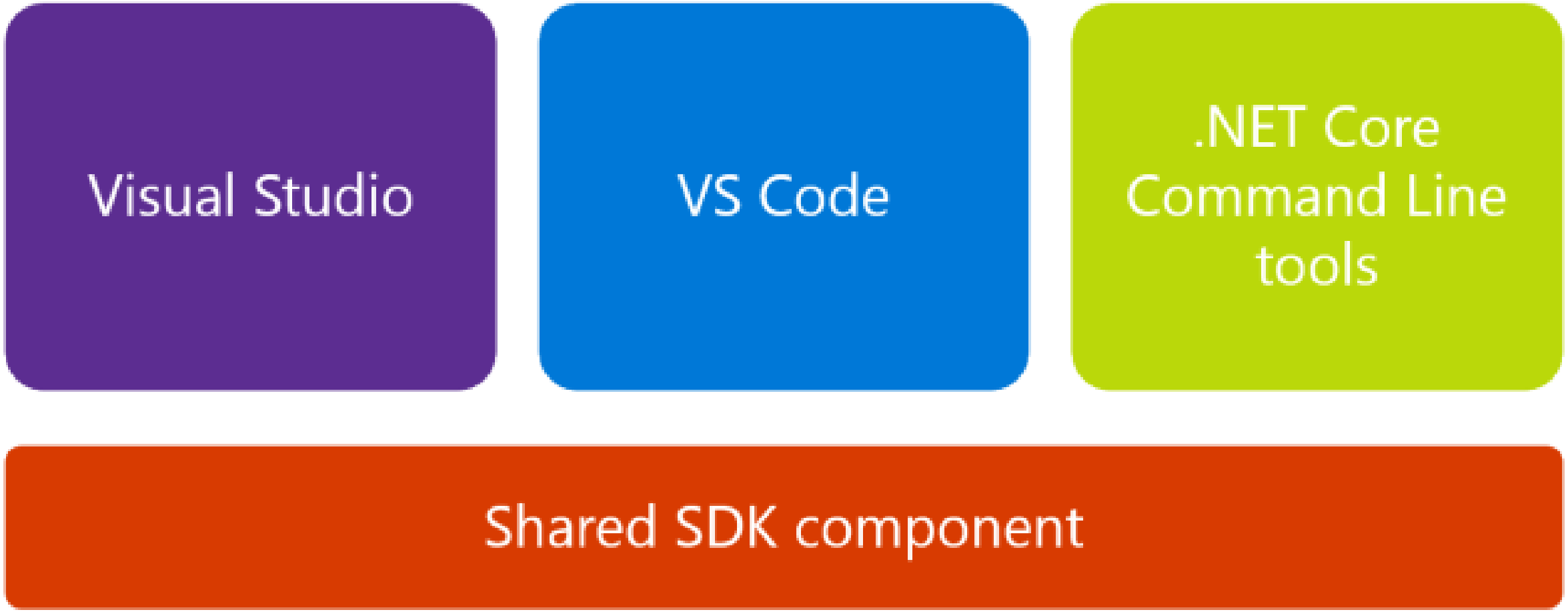
9 lines (really 8 lines without blank lines)

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


Visual Studio
VS Code
.NET Core Command Line tools
Shared SDK component

If you use Visual Studio… you don’t have to worry about the dotnet SDK

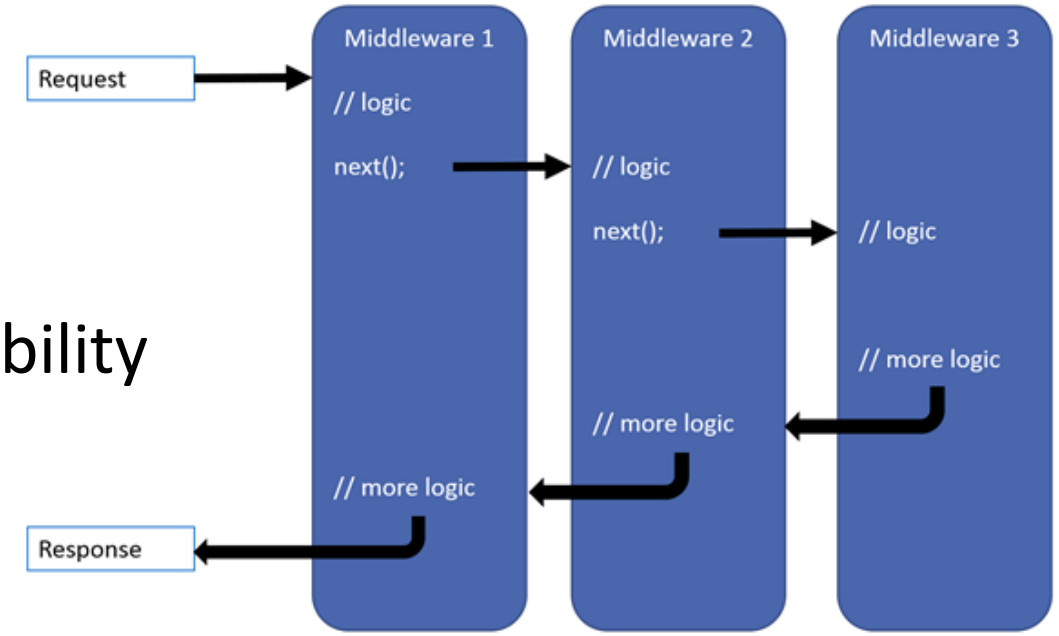
… it’s used by Visual Studio behind the scenes

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
- Environments are a first class citizen
- TagHelpers > Html Helpers
  - HtmlHelper: `@Html.ActionLink("Home", "Index", "Home")`
  - TagHelper: `<a class="btn btn-primary" asp-controller="Home" asp-action="Index">Home</a>`

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
- HTML Helpers still exist
  - But you should use Tag Helpers

Kestrel

- Brand new web server
- Built on libuv
  - …for now
- 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 2 years old
- Barry Dorrans talk on Kestrel’s Security (4:55 to 6:35)

Kestrel with Reverse Proxy

IIS and ASP.NET 4.x

IIS and ASP.NET Core

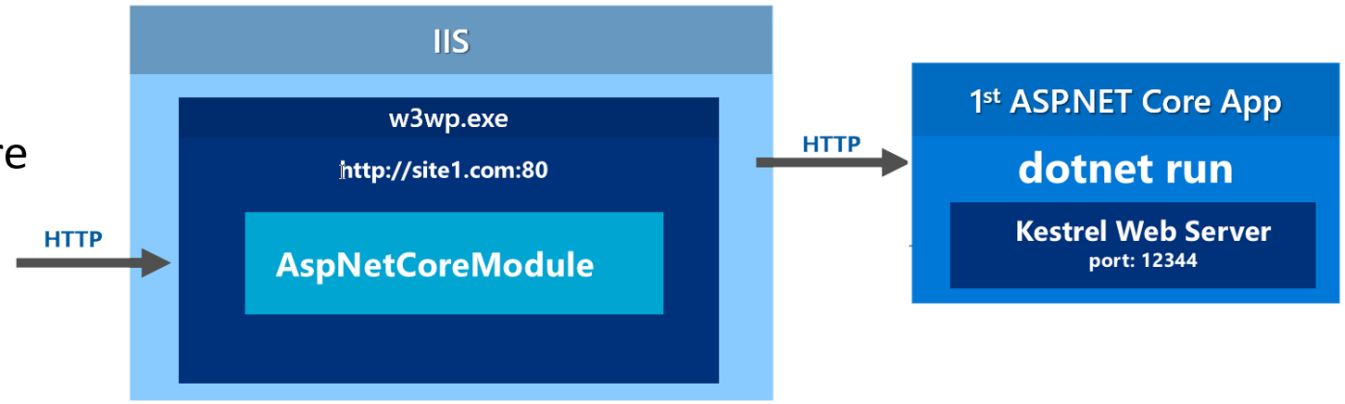


Image: Rick Strahl

Kestrel (continued again)

- High performance
- The stuff they do is crazy in the name of perf
  - More details: https://vimeo.com/172009499

TechEmpower Benchmarks

- What are the TechEmpower Benchmarks?
- https://www.techempower.com/benchmarks/
- ASP.NET Core 1.7M requests per second
  - PHP: 44K
  - Django: 88K
  - Node: 429K
- 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
- Configuration and Logging moved up to Program.cs
- dotnet CLI changes
  - Restore implied
  - 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:

```
public class Program
{
    0 references
    public static void Main(string[] args)
    {
        var host = new WebHostBuilder()
            .UseKestrel()
            .UseContentRoot(Directory.GetCurrentDirectory())
            .UseIISIntegration()
            .UseStartup<Startup>()
            .UseApplicationInsights()
            .Build();

        host.Run();
    }
}
```

ASP.NET Core 2:

```
public class Program
{
    0 references | 0 exceptions
    public static void Main(string[] args)
    {
        BuildWebHost(args).Run();
    }

    1 reference | 0 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" />
    <PackageReference Include="Microsoft.AspNetCore.Mvc" Version="1.0.4" />
    <PackageReference Include="Microsoft.AspNetCore.StaticFiles" Version="1.0.3" />
    <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:

```
<Project Sdk="Microsoft.NET.Sdk.Web">
  <PropertyGroup>
    <TargetFramework>netcoreapp2.0</TargetFramework>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.All" Version="2.0.0" />
  </ItemGroup>
</Project>
```

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) 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
  - Sidenote: if using ASP.NET Core MVC – register the AutoValidateAntiForgeryToken attribute to your global filters

My opinion on Razor Pages

- Love the concept of Page-focused vs. Controller focused
- Feels a bit early
- Idiomatic Razor Pages still needs to be fleshed out
  - Putting View Model props inline on PageModel or still separate ViewModel file and single ViewModel prop on PageModel
    - I lean the latter currently.
- No attribute routing
  - Coming
- I'd wait and see unless you like being on the edge
  - Good news here is – most logic remains in tact between MVC => Razor Pages, with just the few caveats I've shown
- That said… I'm moving to it

Small, but nice

- EnvironmentTagHelper include and exclude

ASP.NET Core 1

```
<environment names="Development">
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</environment>
<environment names="Staging,Production">
    <link rel="stylesheet" href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"
          asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
          asp-fallback-test-class="sr-only" asp-fallback-test-property="position" asp-fallback-test-value="absolute" />
    <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
</environment>
```

ASP.NET Core 2

```
<environment include="Development">
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</environment>
<environment exclude="Development">
    <link rel="stylesheet" href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"
          asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
          asp-fallback-test-class="sr-only" asp-fallback-test-property="position" asp-fallback-test-value="absolute" />
    <link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true" />
</environment>
```

this is == !Development

What's still coming

- 2.1 coming in ~6 months
  - Formal roadmap coming soon.
  - SignalR will be part of 2.1
    - SignalR alpha released on 9/14/17
  - Host in process in IIS, no more out of process (still a reverse proxy)
  - HTTP/2 all the way down to Kestrel
  - KodKod – Web API defaults, codegen for clients (!!!)
  - STS Server
  - Routing available in Middleware via Dispatchers
  - Web API Auth in Templates
  - Kicker – Distributed Systems. Logging, Config, Health Checks, Service Discovery, etc.
- Improve "Developer Inner Loop" to sub-second
- Global tools
  - Ex – Add EF Tools globally instead of needing it in every project
  - Like npm install –g/yarn global add
- .NET IL Linker coming to trim your DLL fat
  - https://github.com/dotnet/core/blob/master/samples/linker-instructions.md

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”
- It’s at least worth a hard look at this point
- 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
- Don't forget to fill out evals, please

Thanks for coming!