Getting Started with ASP.NET Core

J. Tower, Trailhead Technology Partners

CREATED FOR BRILLIANCE AUDIO

About Me

Jonathan "J." Tower
Principal Consultant & Partner
Trailhead Technology Partners

Microsoft MVP in ASP.NET

jtower@trailheadtechnology.com
trailheadtechnology.com/blog
jtowermi

Introductions

What's your name?

What is the focus of your job? What technologies do you use?

Why are you passionate about in software?

What have you heard/read about .NET Core and ASP.NET Core?

What do you want out of this training? Anything you're hoping to avoid?

Schedule

Morning

Introduction

ASP.NET Core Basics

- Morning Break -

ASP.NET Core Basics (continued)

- 60-Minute Lunch Break -

Afternoon

MVC & Web API in Core

- Afternoon Break -

Advanced Topics

Questions

Please, interrupt with any questions

Make sur you get what you want out of the training

If it makes sense, let’s go off plan/try some new thing together

History of .NET

Early History of .NET

Replace Win32 APIs

Next Generation Windows Services (NGWS)

Anders Hejlsberg leaves Borland for Microsoft in 96

COOL: C-like Object Oriented Language

“.NET” because of its embrace of the internet

.NET vs Java

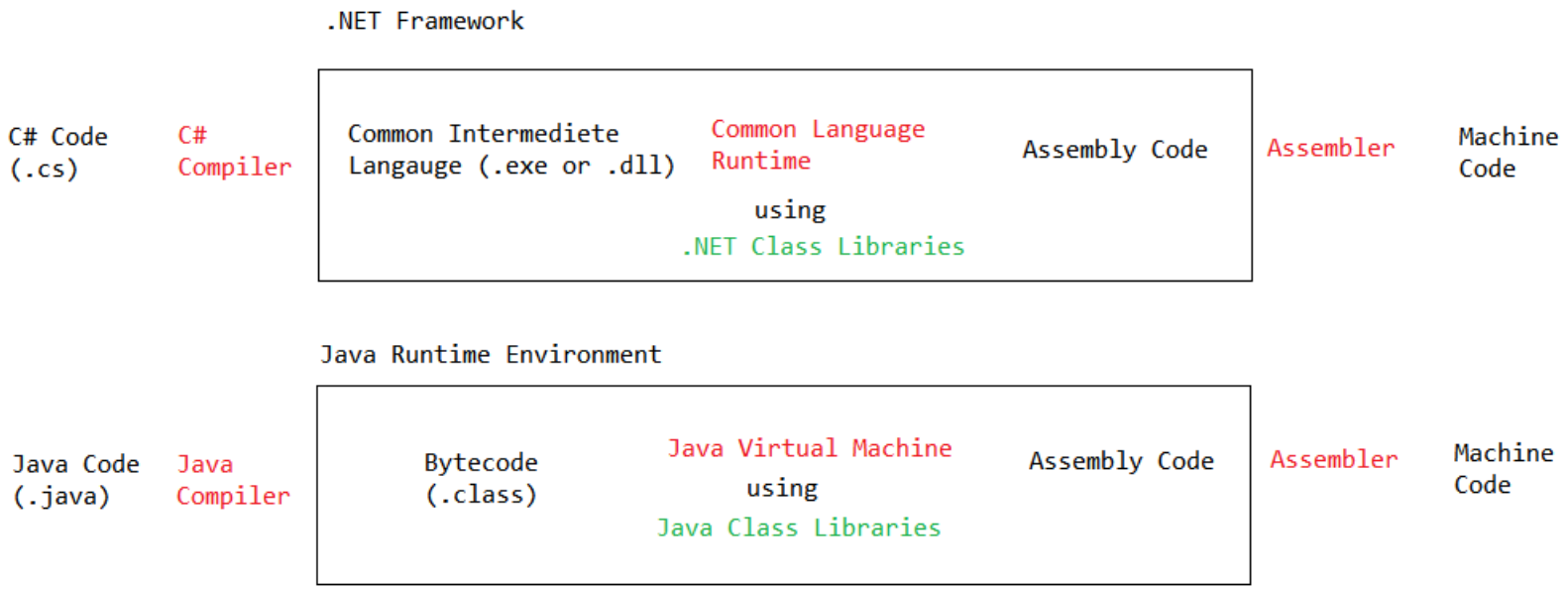
Engineered to be platform-agnostic

Cross-platform implementations available for other OSs

But, Microsoft never implemented full framework outside of Windows

Others did: Mono

.NET vs Java

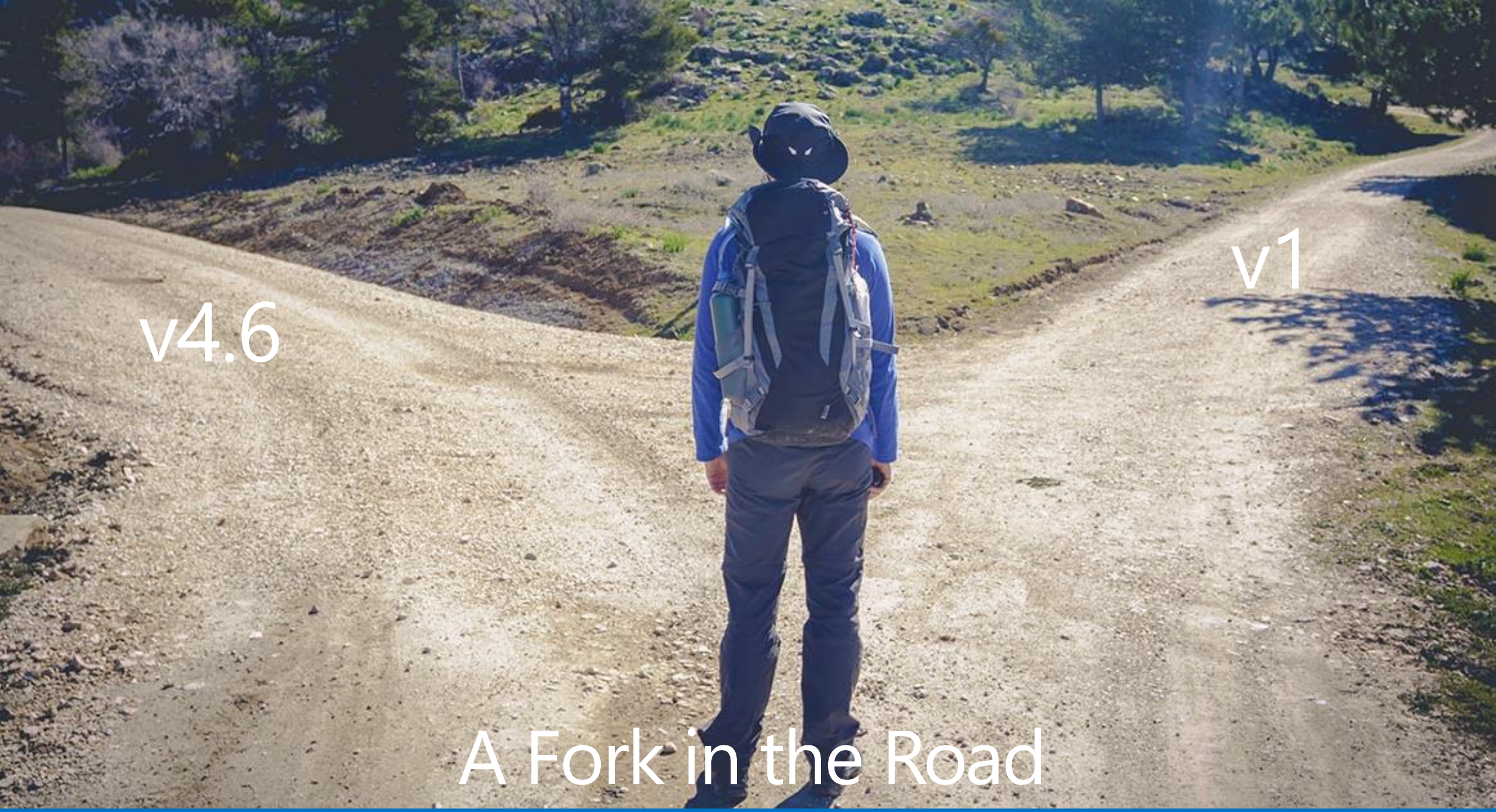

.NET Framework
C# Code
(.cs)
C#
Compiler
Common Intermediete
Langauge (.exe or .dll)
Common Language
Runtime
using
.NET Class Libraries
Assembly Code
Assembler
Machine
Code
Java Runtime Environment
Java Code
(.java)
Java
Compiler
Bytecode
(.class)
Java Virtual Machine
using
Java Class Libraries
Assembly Code
Assembler
Machine
Code

Cross-Platform .NET:
A Promise 15 Years in the Making

History of ASP.NET

History of ASP.NET

Version	.NET / IDE Version	Key Features
1.0	1.0 and Visual Studio .NET	OO, event-driven web apps, viewstate, DLL class libraries
1.1	1.1 and Visual Studio .NET 2003	ODBC
2.0	2.0 and Visual Studio 2005	New controls, Themes, Skins, Master pages, Membership Services, Localization
3.5	3.5 and Visual Studio 2008	ASP.NET AJAX, LINQ, **MVC 1**
4.0	4.0 and Visual Studio 2010	Routing
4.5	4.5 and Visual Studio 2012	Unobtrusive Validation, Async, HTML5, WebSocket
4.5.1	4.5.1 and Visual Studio 2013	One ASP.NET, Scaffolding, Identity


v4.6
v1
A Fork in the Road

Wait, I Thought It Was ASP.NET 5

Well, about that…

ASP.NET 4.6 and ASP.NET Core 1.0

ASP.NET 4.6	~~ASP.NET 5~~
.NET Framework 4.6	.NET Core 1.0
.NET framework libraries	.NET core libraries
Compilers and runtime components (.NET Compiler Platform: Roslyn, C#, VB, F# Languages, RyuJIT, SIMD)	

Source: http://www.hanselman.com/blog/ASPNET5IsDeadIntroducingASPNETCore10AndNETCore10.aspx

ASP.NET 4.6 and ASP.NET Core 1.0

ASP.NET 4.6	ASP.NET Core 1.0
.NET Framework 4.6	.NET Core 1.0
.NET framework libraries	.NET core libraries

Compilers and runtime components
(.NET Compiler Platform: Roslyn, C#, VB, F# Languages, RyuJIT, SIMD)

Source: http://www.hanselman.com/blog/ASPNET5IsDeadIntroducingASPNETCore10AndNETCore10.aspx

ASP.NET 5, We Hardly Knew Ye

ASP.NET 5 now becomes **ASP.NET Core .10**

EF 7 becomes **EF Core 1.0**

Core 1.0 – because it's new, different, and built for .NET Core

ASP.NET – because it's still so familiar

.NET Core Roadmap

Release	Time frame*
1.0.1	Sept 13, 2016
1.1	Q4 2016 / Q1 2017
1.2	Q1 2017 / Q2 2017

1.1

Replacing xproj/project.json
with .csproj/MSBuild

1.2

Bring back many of the missing APIs in .NET Core:

- Networking
- Serialization
- Data
- Much more

ASP.NET Core Roadmap

Release	Time frame*
1.1	Q4 2016 / Q1 2017
1.2	Q1 2017 / Q2 2017

1.1
WebSockets
URL Rewriting middleware
Response caching middleware
DI improvements for 3rd party containers
WebListener server (Windows only)
Middleware as MVC filters
ViewComponents as Tag Helpers
Improved Azure integration
App Service startup time improvements
App Service logging provider
Azure Key Vault provider

1.2
SignalR
View Pages (Views without MVC controllers)
Web API security
View precompilation

Future Work
Visual Basic support

DEMO: ASP.NET Core on Windows

Hello, World!

DEMO: ASP.NET Core on MacOS

Hello, Mac!

LAB: Setup and Installation

Windows

Visual Studio 2015 Update 3

VS 2015 Tooling Preview 2

https://www.microsoft.com/net/core#windows

MacOS

Homebrew

OpenSSL

.NET Core SDK

VS Code

https://www.microsoft.com/net/core#macos

Why ASP.NET Core?

Reasons for ASP.NET Core

Cross-Platform Deployment/Development

Unified stack for UIs and APIs

Support for moderns client-side frameworks

Light-weight and modular HTTP request pipeline

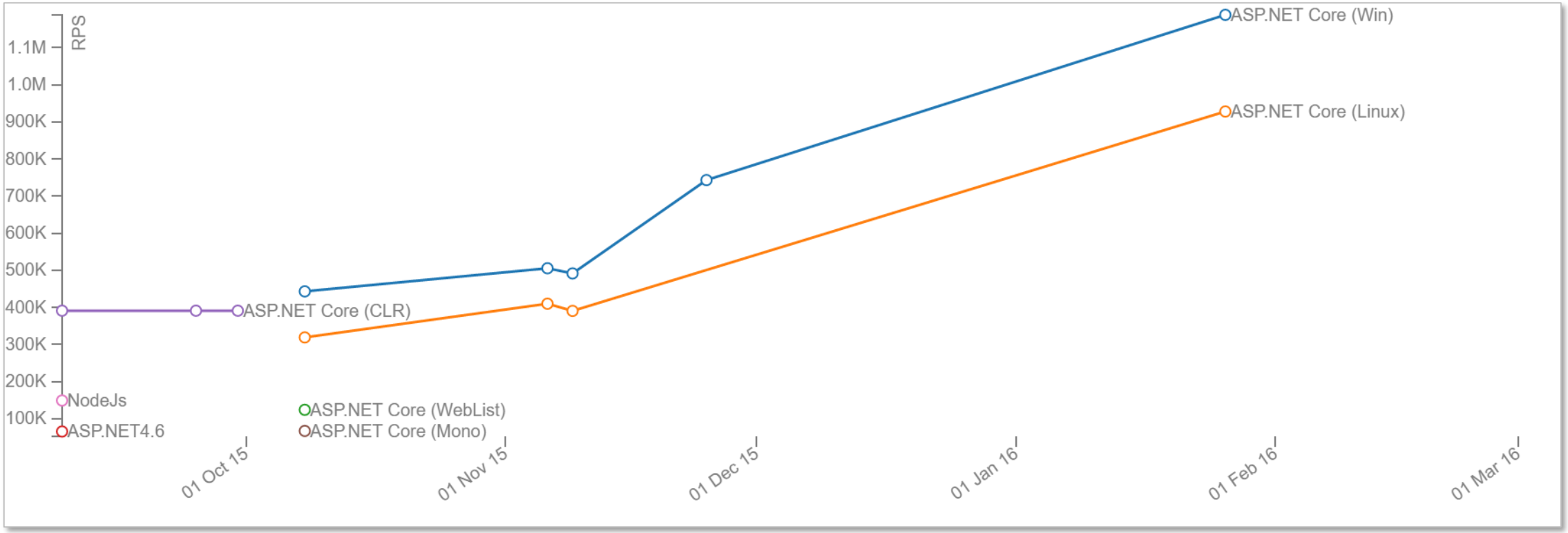
Self hosting

Side-by-side app versioning

Open source and community focused

Performance

Perf!


RPS
1.1M
1.0M
900K
800K
700K
600K
500K
400K
300K
200K
100K
ASP.NET Core (Win)
ASP.NET Core (Linux)
ASP.NET Core (CLR)
NodeJs
ASP.NET4.6
ASP.NET Core (WebList)
ASP.NET Core (Mono)
01 Oct 15
01 Nov 15
01 Dec 15
01 Jan 16
01 Feb 16
01 Mar 16

New Project Templates

DEMO: New Project Templates

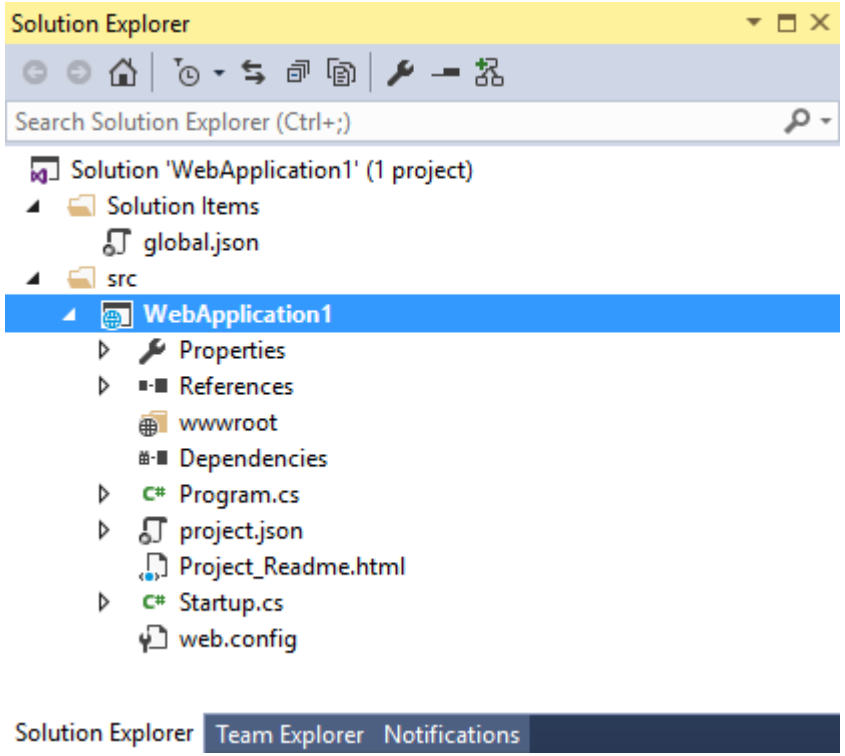
VS, File > New Project…

dotnet new

yeoman

Project & Solution Structure

Project & Solution Structure



project.json File

Maintain dependencies in your project (package.config)

Toolings support

Target frameworks

Assembly Information

ProjectName.xproj File

Contains Visual Studio-specific settings

Getting renamed back to .csproj

Future versions will see build settings moved to .csproj

project.json File

Contains all project build settings (name, assembly version, etc.)

Also contains NuGet dependencies

Getting phased out and/or renamed in future releases

- Build-related settings moving back to .csproj
- May be renamed to nuget.json and retain nuget dependencies

project.lock.json File

Generated by the .NET tooling when you restore the project's packages

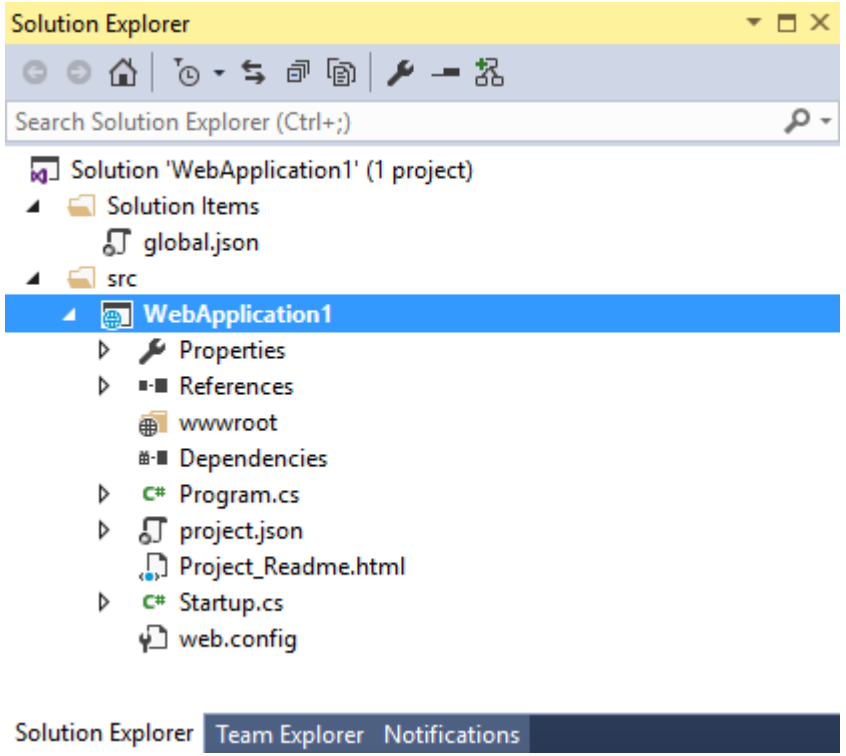
Cache specific dependency graph

Regenerated when project.json editted

Don’t edit it or check it into source control

Edit project.json directly

wwwroot Folder



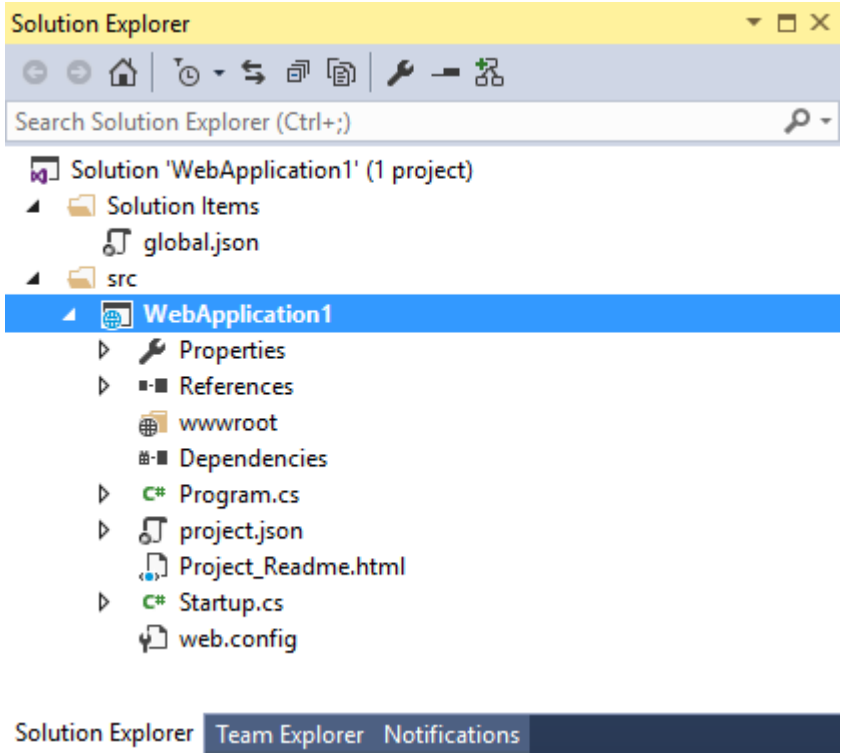
For static and generated files

Separates source code from distributed site

Better security

NOTE: To enable Static Files feature, add NuGet package and configure in Configure() method by adding app.UseStaticFiles().

Dependencies Section



Used to manage Bower and NPM tooling packages

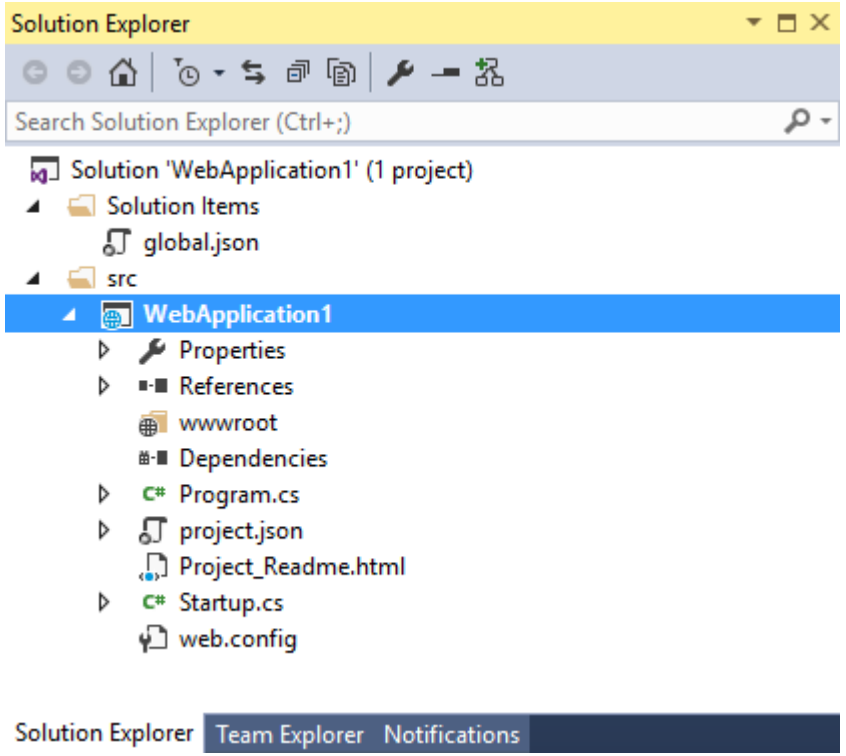
For client-side packages (i.e. jQuery, Bootstrap, Angular, Lodash, etc).

bower.json file

package.json file

NOTE: Bower and NPM are not tied to ASP.NET Core, they are open source technologies which can be used in ASP.NET Core.

Program.cs File



The most important file in ASP.NET Core

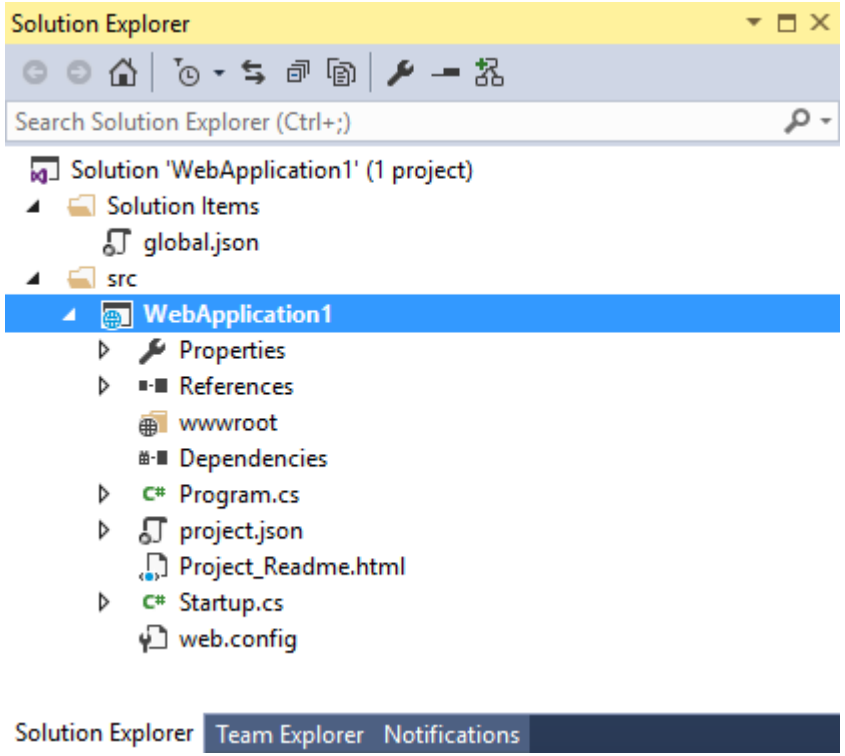
Contains a Main() method like a console App

ASP.NET Core 1.0 applications are console applications with a well-defined entry point

Initiates, builds, and runs the server (IIS and Kestrel)

Define the root of the project here

Startup.cs



Second most important file

ConfigureServices() and Configure()

Configure middleware on the request pipeline

Configure Dependency Injection (DI)

Other start-up configuration

Startup.Startup (Contructor)

For loading app configuration

Startup.ConfigureServices()

Called first after contructor

Add services to ASP.NET here via DI

Add your own services via DI to

Service config often wrapped in extension methods

Startup.Configure()

Runs second after constructor

Configures the ASP.NET pipeline (default: empty)

Web.Config File

For IIS support

Simple HttpModule that passes requests on to Kestrel

```
<handlers>
   <add name="aspNetCore" path="*" verb="*"
        modules="AspNetCoreModule" resourceType="Unspecified"/>
</handlers>
```

DEMO: gulpfile.js file

A “task runner” for front-end web development

gruntfile.js File

Another task run option

The community seems to be moving toward gulp and home-made

bundleconfig.json File

appsettings.json File

No more:

- XML
- System.Configuration namespace
- <AppSettings> in web.config
- <ConnectionStrings> in web.config

JSON-format by default (also support XML, INI, and custom)

appsettings.json File

```
var builder = new ConfigurationBuilder()
    .SetBasePath(env.ContentRootPath)
    .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
    .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true);

if (env.IsDevelopment()) builder.AddUserSecrets();

builder.AddEnvironmentVariables();
Configuration = builder.Build();
```

DEMO: Config Environment Variables

DEMO: User Secrets

Saved on Windows:
"%APPDATA%\microsoft\UserSecrets\<userSecretsId>\secrets.json"

In project.json
Set a "userSecretsId"
Add "Microsoft.Extensions.SecretManager.Tools"
Add "Microsoft.Extensions.Configuration.UserSecrets"

Command Line:
dotnet user-secrets list
dotnet user-secrets set key value
dotnet user-secrets remove key
dotnet user-secrets clear

Visual Studio Editor

DEMO: Add a Setting File

LAB: Add a Settings File to Project

Add a new settings file

Parse the setting into an object

Inject the setting object in a controller

Use the setting in the controller

Hosting ASP.NET Core

Hosting ASP.NET Core

Totally decoupled from web server environment

Supports:

- IIS / IIS Express
- Self-Hosting using Kestrel and WebListener HTTP servers
- Third party implementations

Kestrel

Cross-platform web server

Based on libuv, a cross-platform asynchronous I/O library

Microsoft.AspNetCore.Server.Kestrel

Meant to be deployed behind a proxy (like IIS, Apache, or Nginx)

Classic ASP.NET Hosting

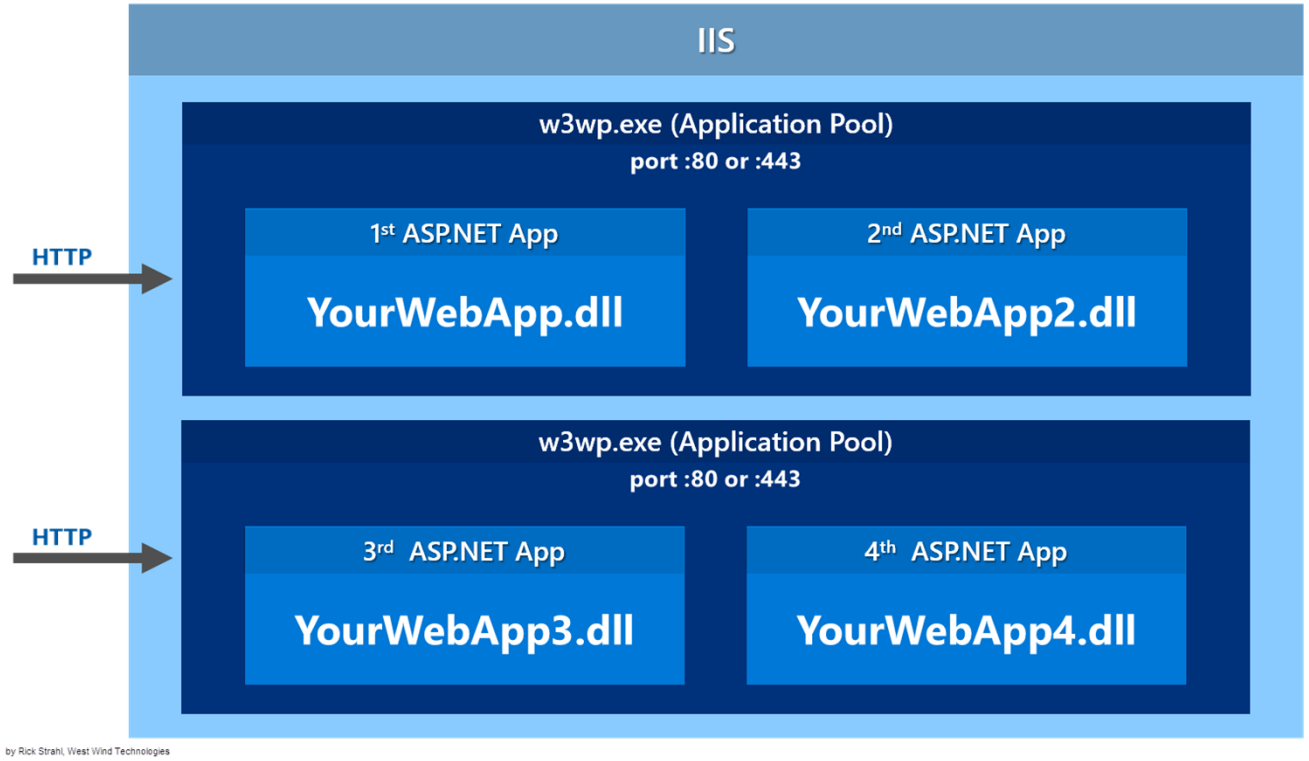


Photo credit: https://weblog.west-wind.com/posts/2016/Jun/06/Publishing-and-Running-ASPNET-Core-Applications-with-IIS

IIS and Kestrel

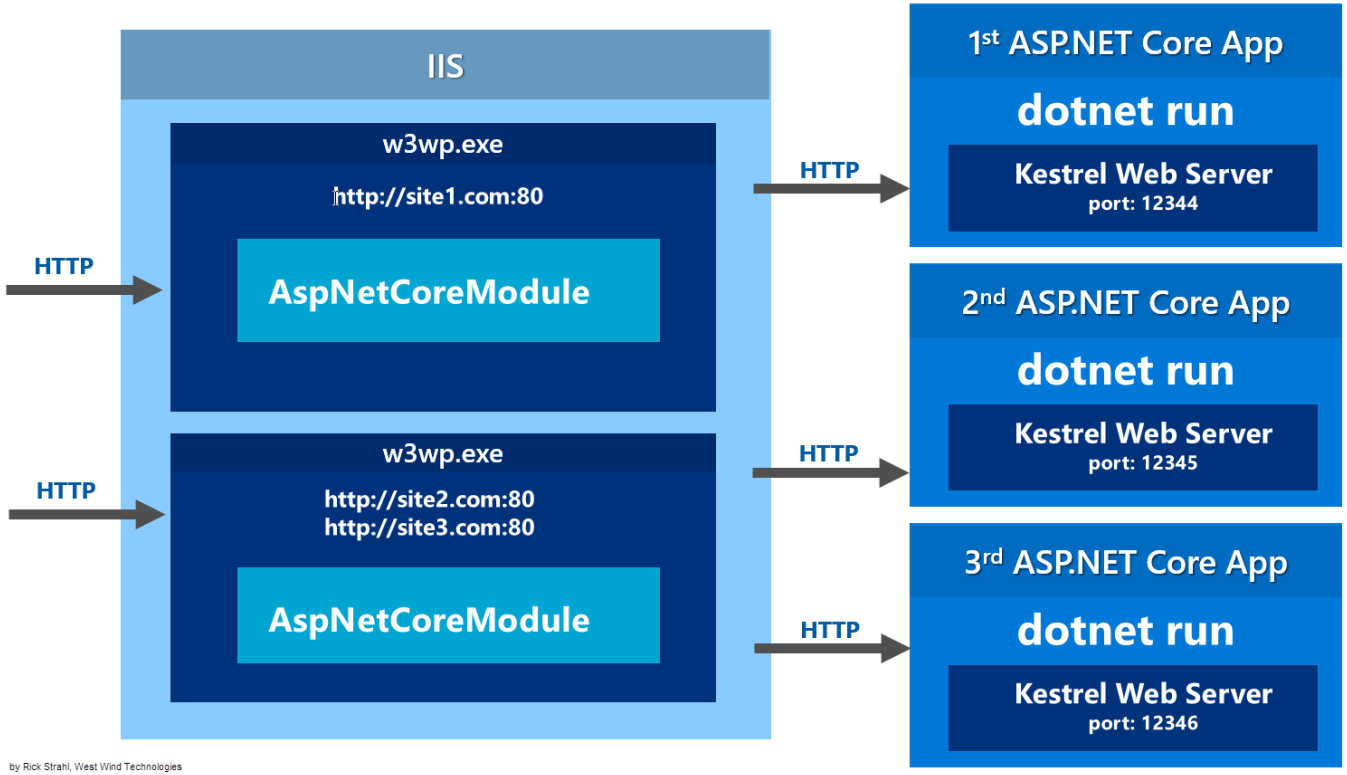
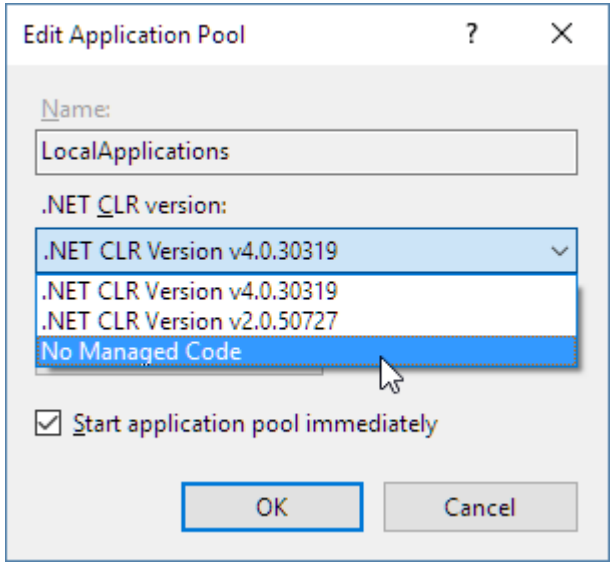


Photo credit: https://weblog.west-wind.com/posts/2016/Jun/29/First-Steps-Exploring-NET-Core-and-ASPNET-Core

IIS / IISExpress

Windows-only

No need in dev environment



WebListener

Windows-only

Runs directly on Http.sys

Still in preview!

Dependency Injection

Dependency Injection Lifetimes

Transient

Transient lifetime services are created each time they are requested. This lifetime works best for lightweight, stateless services.

Scoped

Scoped lifetime services are created once per request.

Singleton

Singleton lifetime services are created the first time they are requested (or when ConfigureServices is run if you specify an instance there) and then every subsequent request will use the same instance. If your application requires singleton behavior, allowing the services container to manage the service's lifetime is recommended instead of implementing the singleton design pattern and managing your object's lifetime in the class yourself.

DEMO: Injecting a Service with DI

LAB: Create your own DI Service

Create a service that mimics sending and email

Create an interface for that service

Map the interface to the service instance

Inject an instance of your service into a controller

EXTRA: Try injecting different services for different environments

Package Management

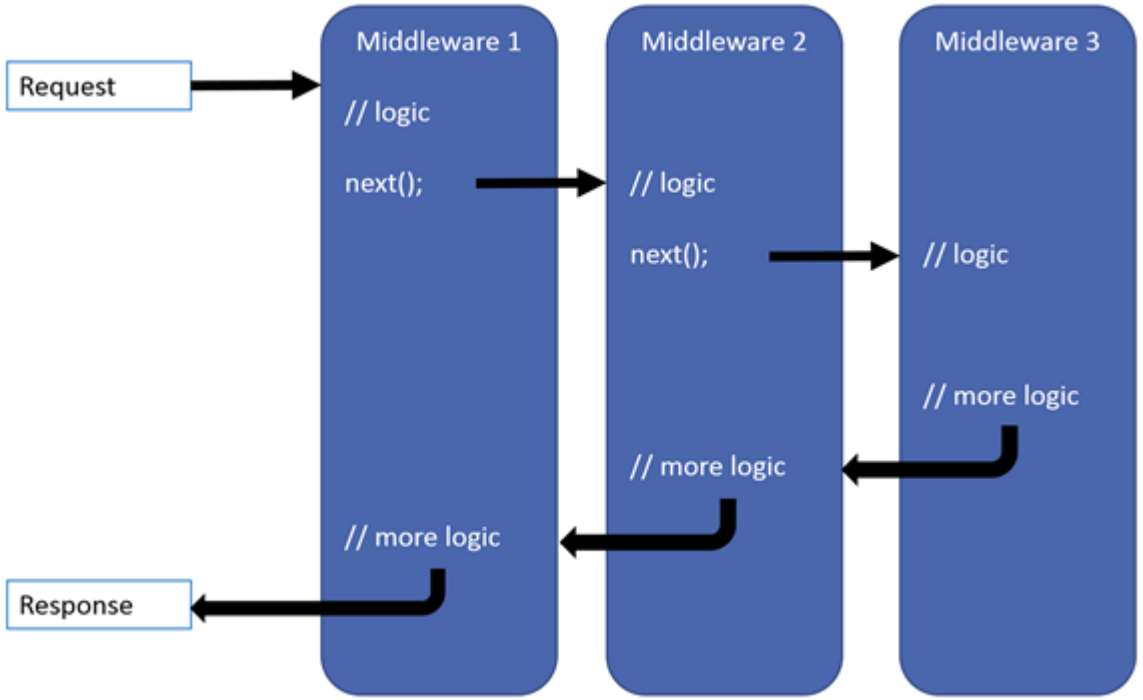
DEMO: Loading packages

Server-Side/.NET
- NuGet

Client-Side
- NPM
- Bower

Pipeline & Middleware

Pipeline & Middleware



Built-In Middleware

Middleware	Description
Authentication	Provides authentication support
CORS	Configures Cross-Origin Resource Sharing.
Routing	Define and constrain request routes.
Session	Provides support for managing user sessions.
Static Files	Provides support for serving static files, and directory browsing.

LAB: Create Custom Middleware

MVC & APIs

Setting up MVC

1. File > New Project, ASP.NET Core, Empty
2. Microsoft.AspNetCore.Mvc and Microsoft.AspNetCore.StaticFiles
3. `services.AddMvc();`
4. `app.UseStaticFiles();`
5.
```
app.UseMvc(config =>
{
    config.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
});
```
6. Add Controller and View

Setting up WebAPI

No additional steps!

Same references, base class, and pipeline as MVC

Unified Controllers

Shared routes and route table

Shared security pipeline

Web API no longer supports inferred verbs

API and View actions in the same controller

Return IActionResult for views and APIs
(no need for IHttpActionResult for APIs)

Unified Controllers

MVC Controller

```
public MyController : System.Web.MVC.Controller { public IActionResult MyAction() {} }
```

Web API Controller

```
public MyController : System.Web.Http.ApiController { public IHttpActionResult MyAction() {} }
```

Unified Controller

```
public MyController : Microsoft.AspNet.Mvc.Controller { public IActionResult MyAction() {} }
```

DEMO: View API Controllers

LAB: Create View and API Controllers

Create an MVC view controller and view

Create a layout view

Test your view

Create an API controller that returns hard-coded data

Create another API action that returns an error

Test your API controller

Tag Helpers

Tag Helpers

An HTML-friendly development experience

```
@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizo
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}
```

```
<form asp-controller="Account" asp-action="Register" method="post" class="form-hori
    <h4>Create a new account.</h4>
    <hr />
    <div asp-validation-summary="ValidationSummary.All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="Email" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="Email" class="form-control" />
            <span asp-validation-for="Email" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Password" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="Password" class="form-control" />
            <span asp-validation-for="Password" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <label asp-for="ConfirmPassword" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <input asp-for="ConfirmPassword" class="form-control" />
            <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <button type="submit" class="btn btn-default">Register</button>
        </div>
    </div>
</form>
```

Tag Helpers

A rich IntelliSense environment for creating HTML and Razor markup

```
<label asp-for="e"
```

```
string RegisterViewModel.Email { get; set; }
```

- ConfirmPassword
- Email
- Equals
- GetHashCode
- GetType
- Password
- ToString

```
<input asp-for="Email" cl
```

- class
- onclick
- onclose
- ondblclick

Tag Helpers

A way to make you more productive and able to produce more robust, reliable, and maintainable code using information only available on the server

```
<link rel="stylesheet" href="~/css/site.min.css" asp-append-version="true"/>

<link rel="stylesheet"
    href="/css/site.min.css?v=UdxKHVNJA5vb1EsG9O9uURFDfEE3j1E3DgwL6NiDGMc" />
```

DEMO: Some Build-In Tag Helpers

Environment

Form

Anchor

Cache

Image

DEMO: Custom Tag Helper

LAB: Custom Tag Helpers

Create a custom ProgressBar tag helper

- Create a new ASP.NET Core project
- Add bootstrap using Bower or NPM

View Components

View Components

Similar to partial views

Doesn’t use model binding

Avoids controller lifecycle (like action filters)

Not reachable via HTTP

Uses parameters and DI to get data

DEMO: View Components

Entity Framework Core 1.0

Entity Framework Core 1.0

Only available on .NET Core

Cross-Platform

In-memory provider

Modular (NuGet)

More async support

NOTE: Not (yet) feature compatible with EF 6.x

DEMO: EF Core Code-First & Migrations

LAB: EF Core Setup and Migration

Add references for EF tools and dependencies

Create a context and model classes

Configure your context as a service

Add a database migration

Update the database

Command-Line Interface (CLI)

.NET CLI

```
> dotnet new
> dotnet restore
> dotnet build
> dotnet publish
> dotnet run
> dotnet test
> dotnet pack
```

dotnet restore

Specify sources

- Globally: C:\users\username\AppData\Roaming\NuGet\NuGet.config
- Locally: NuGet.config or --configfile
- Command Line: --source switch

Set package output directory

- --packages new-location
- Default: %userprofile%\.nuget\packages

dotnet build

Some important flags

- --output
- --framework
- --runtime
- --configuration

dotnet publish

Framework-dependent deployment (FDD)

```
"type": "platform"
> dotnet app.dll
```

Self-contained deployment (SCD)

```
//"type": "platform"
> app.exe
```

dotnet run

Run at development time

Run FDDs

Sample flags

--configuration

-- project /path/to/project.json

dotnet pack

```
> dotnet pack
```

--no-build

--version-suffix “20161217“

```
1.0.0-*
1.0.0-20161217
```

ASP.NET Identity

ASP.NET Identity

Install package
Microsoft.AspNetCore.Identity.EntityFrameworkCore

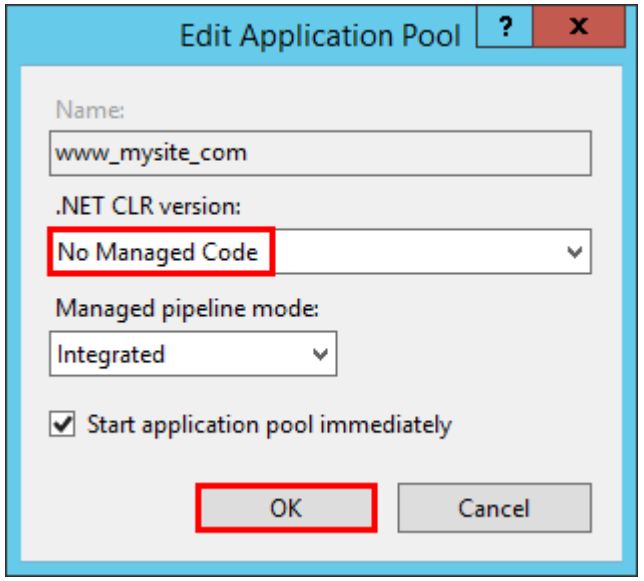
```
services.AddIdentity<ApplicationUser, IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();

app.UseIdentity();
```

Deploying ASP.NET Core Applications

Deploying ASP.NET Core to IIS

Ensure *.UseIISIntegration()*

Install the .NET Core Windows Server Hosting bundle



Other Common Scenarios

Azure / AWS
https://docs.asp.net/en/latest/tutorials/publish-to-azure-webapp-using-vs

Docker
https://azure.microsoft.com/en-us/documentation/articles/vs-azure-tools-docker-hosting-web-apps-in-docker/

Linux with Nginx or Apache (or other)
https://docs.asp.net/en/latest/publishing/linuxproduction

Thank You!

Jonathan "J." Tower
Principal Consultant & Partner
Trailhead Technology Partners

Microsoft MVP in ASP.NET

jtower@trailheadtechnology.com
trailheadtechnology.com/blog
jtowermi

TRAILHEAD
TECHNOLOGY PARTNERS
trailheadtechnology.com