Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS | San Diego

ASP.NET Core for Mere Mortals

Philip Japikse (@skimedic)

MVP, ASPInsider, MCSD, CSM, PSM, PSD

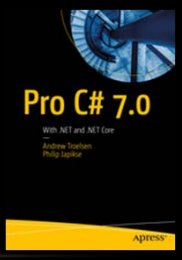
Architect, Author, Speaker

Code Again for the First Time!

Visual Studio 25 YEARS OF CODING INNOVATION

Phil.About()

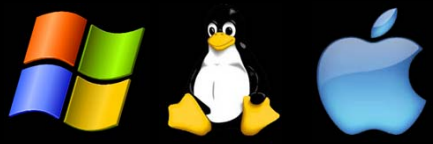
- Consultant, Coach, Author, Teacher
  - Lynda.com (http://bit.ly/skimediclyndacourses)
  - Apress.com (http://bit.ly/apressbooks)
- Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, PSM, PSD
- Founder, Agile Conferences, Inc.
  - http://www.dayofagile.org
- President, Cincinnati .NET User's Group



.NET CORE

WHAT IS .NET CORE?

- Rewrite of “full” .NET Framework
- Vast performance improvements over prior versions
  - Including native compilation
- Flexible deployment model
  - Windows, Linux, Mac

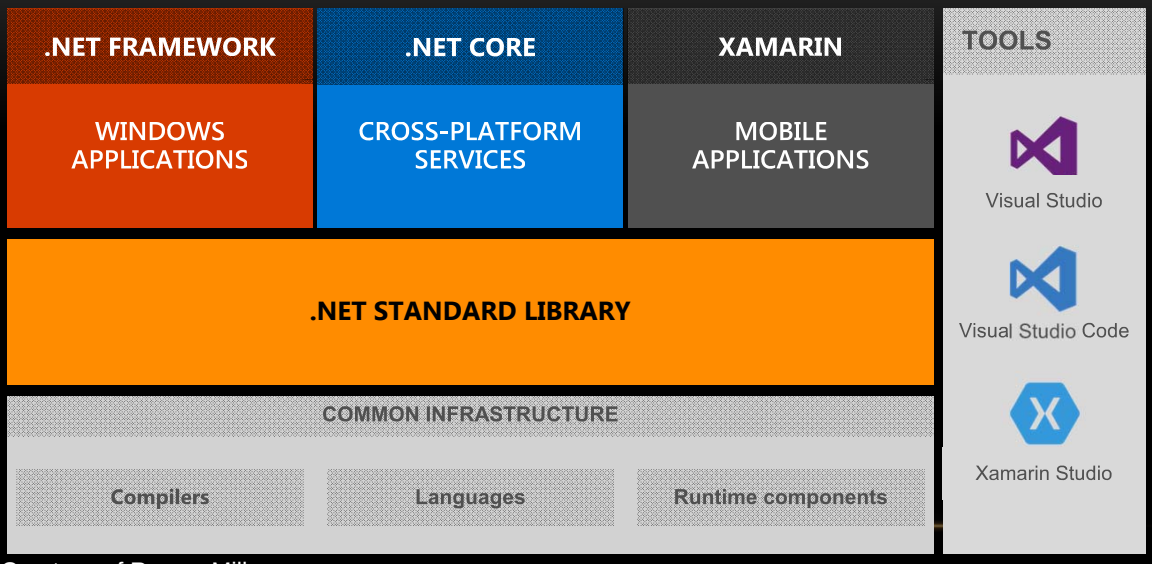


- Full command line support
- True side by side installation support
- Open source from the start
  - Many improvements and features provided by the community

ANATOMY OF A .NET CORE APPLICATION (BOTTOM TO TOP)

- .NET Core Runtime (CoreCLR) - GC, JIT Compiler, base .NET Types
- .NET Core Framework Libraries (CoreFX) - Mostly platform/OS agnostic
- Application Host (dotnet.exe) and Command Line Interface (CLI)
- Custom Applications - Console Apps or Class libraries

FULL BCD (BIRTHDAY CAKE DIAGRAM)



Courtesy of Rowan Miller
https://github.com/rowanmiller/Demo-EFCore

DEPLOYMENT

- Deployment models
  - Self contained –includes .NET Core f/w
  - Portable – expects .NET Core installed on deployment machine
- Kestrel adds a layer of complexity – see the docs

WHAT'S NEW IN 2.0

- .NET Standard 2.0
- 6 new platforms supported
  - Can target Linux as "single" OS
- .NET Core can reference .NET F/W Packages and Projects
- "dotnet restore" is now implicit
- Performance Improvements
- .NET Standard 2.0 NuGet Packages
- Limited Visual Basic support
- Live Unit Testing .NET Core 2
- Docker updates

WHAT'S NEW IN .NET CORE 2.1 PREVIEW

- .NET Core Global Tools
  - Inspired by NPM Global Tools
- Minor Version Roll-Forward
- Span<T>, Memory<T>
- Windows Compatibility Pack
  - 20K more APIs on windows
- Many more supported platforms
- Performance Improvements
  - Build
  - Sockets
- Elimination of dependencies on:
  - WinHTTP/libcurl
- Docker Improvements
  - Smaller images, faster updates

.NET CORE SUPPORT LIFECYCLES

- Long Term Support (LTS)
  - Major releases (e.g. 1.0, 2.0)
  - Only upgraded with critical fixes (patches)
  - Supported for three years after GA release or at least one year after the next LTS release.
  - NOTE: 1.1 was added to the LTS list with the release of 2.0
  - 2.1 as added to the LTS list when it was released
- Current
  - Minor releases (e.g. 1.1, 1.2)
  - Upgraded more rapidly
  - Supported for three months after next Current release

https://www.microsoft.com/net/core/support

ASP.NET CORE 2.X

ASP.NET CORE

- ASP.NET Core is ASP.NET rebuilt on top of .NET Core
- Single, cross-platform framework for web, services, and microservices
  - WebApi + MVC + Web Pages + Razor Pages = ASP.NET Core
- Takes advantage of .NET Core performance
  - Includes a high performance web server (Kestrel) built on LibUV

ASP.NET CORE FEATURES

- Pluggable Middleware - Routing, authentication, static files, etc.
- Full Dependency Injection integration
- Simplified and Improved Configuration System
- Tag Helpers
- View Components

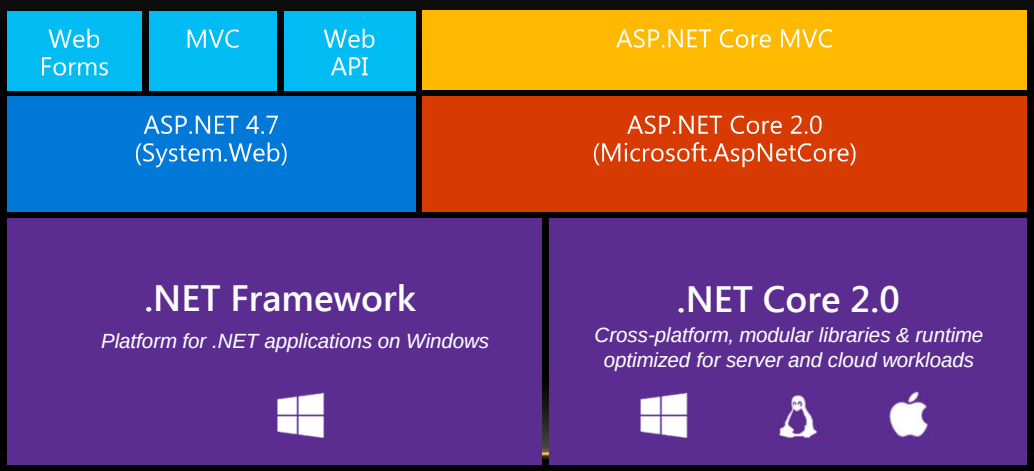
WHAT'S NEW IN ASP.NET CORE 2.0

- Razor Pages
- Updated Templates
  - Razor pages, Angular, React
- DbContext Pooling with EF Core 2.0
- Razor support for C# 7.1
- Simplified configuration and startup
- Microsoft.AspNetCore.All metapackage
  - Includes EF SQL Server as well

WHAT'S NEW IN ASP.NET CORE 2.1

- SignalR
- WebHooks
  - 12-14 Receivers from ASP.NET
- Partial Tag Helpers
- MVC Improvements
  - Functional testing "ootb"
  - Razor improvements
- Identity as a package
  - Or Scaffold into existing app
- Web API Improvements
  - Enhanced Controllers
- Improvements for EU – GDPR
  - Hooks in Identity, cookies, encryption
- HTTPS Improvements
  - On by default
  - Cleaner redirect
- HTTP Client Factory
- Microsoft.AspNetCore.App metapackage

ASP.NET CORE BCA



RUNNING .NET CORE APPLICATIONS

RUNNING ASP.NET CORE APPLICATIONS

- Visual Studio
  - Select IIS or Kestrel
  - Port is controlled by launchSetting.json
- .NET Core CLI
  - 'dotnet run'
  - Port defaults to 5000
    - Can be changed using WebHostBuilder

PROJECT FILES

CSPROJ FILE

- Allows for MSBuild support
- Holds package and project references
  - VS 15.3+ shows nodes in Solution Explorer for packages
  - Removal of packages.config
- Full support for file globbing

CONFIGURING THE WEB SERVER(S)

ASP.NET CORE APPS ARE CONSOLE APPS

- Web server(s) is(are) created in Program Main() method

```
var host = new WebHostBuilder()
                .UseKestrel()
                .UseContentRoot(Directory.GetCurrentDirectory())
                .UseIISIntegration()
                .UseStartup<Startup>()
                //Configuration will be discussed soon
                .ConfigureAppConfiguration(hostingContext,config)
                .UseUrls("http://*:40001/") //Configures Kestrel
                .Build();
host.Run();
```

- In ASP.NET Core 2.0, replaced with CreateDefaultBuilder()

LAUNCHSETTINGS.JSON CONTROLS RUNNING APP

- IIS Settings
  - Sets app URL/SSL Port, auth settings
- Profiles (appear in VS Run command)
  - IIS Express
    - Sets environment variable
  - <AppName>
    - Sets URL, environment variable

APPLICATION CONFIGURATION

ENVIRONMENTAL AWARENESS

- ASP.NET Core uses ASPNETCORE_ENVIRONMENT variable
  - Development, Staging, Production are built-in environments
- Environment is used throughout ASP.NET Core
  - Determining which configuration files to load
  - Running different code based on the environment using IHostingEnvironment
  - Environment Tag Helper
- Simplifies deployment and testing

APPLICATION CONFIGURATION

- Applications are configured using:
  - Simple JSON (or other file types)
  - Command line arguments
  - Environment variables
  - In memory .NET objects, Encrypted user store, Custom providers
- Configuration values are set in the order received
- Environment determines which additional file(s) to load
  - appsettings.<environment>.json

APPLICATION CONFIGURATION

- Custom classes can represent configuration values
  - Can bind to entire configuration or individual sections with services.Configure<T>
  - Requires the Microsoft.Extensions.Options.ConfigurationExtensions package
  - Can be added to DI container and injected in with IOptionsSnapshot<T>

THE STARTUP CLASS

APPLICATION STARTUP

- The Startup class configures services and application pipeline
- Constructor creates configuration builder, configures user secrets
- Configure sets up how application will respond to HTTP requests
- ConfigureServices configures services and DI

- Changed in 2.0:
  - Configuration created in DefaultWebHostBuilder, injected into Startup.cs

CONFIGURING EF CORE CONTEXT POOLING

- New feature in ASP.NET/EF Core 2
- Context must have single public constructor that takes DbContextOptions

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContextPool<StoreContext>(options =>
       options.UseSqlServer(Configuration.GetConnectionString("SpyStore")));
}
```

DEPENDENCY INJECTION

BUILT-IN DEPENDENCY INJECTION

- Items added to the services container in Startup.cs
- Services are accessed through:
  - Constructor injection
  - Method injection (with [FromServices])
  - View injection (with @inject)
- Can also retrieve services through:
  - ApplicationServices (for non-controller classes)
  - HttpContext.RequestServices (for controllers)
  - Injection is the preferred mechanism

REGISTER CUSTOM SERVICES

- Custom services can be registered as well:
  - Transient: Created each time they are requested
  - Scoped: Created once per HTTP request
  - Singleton: Max of one instance per application
  - Instance: Similar to singleton, but created when Instance is called

ROUTING

ROUTING

- Attribute Routing is first class citizen in ASP.NET Core
  - Helps to refine routing for individual controller actions
- Route table used for default route
  - Sometimes skipped in ASP.NET Core Service Applications
- Controller and actions can define specific routes
  - If routing added to Controller, must be added to all Actions

CONTROLLERS

CONTROLLERS AND ACTIONS

- Everything derives from single Controller class
  - Controller, AsyncController, APIController all rolled into one
- API methods must specify HTTP Verb - no long based on name of method
- All return IActionResult (or Task<IActionResult>)

- Many helper methods built into base Controller for returning HttpStatusCodes
  - NoContent (204), OK (200), BadRequest (400), etc.

MANAGING CLIENT SIDE LIBRARIES

MANAGING CLIENT SIDE LIBRARIES

- Bower is dead
- Library Manager is coming in VS 15.8
  - https://github.com/aspnet/LibraryManager
  - Download, compile, install VSIX
- Libraries are managed in libman.json
  - Cdnjs is default library source
  - Can be configure per package or globally
- Another great tool by Mads Kristensen

BUNDLING AND MINIFICATION

BUNDLING AND MINIFICATION

- JavaScript and CSS files should be bundled and minified for performance
- VS 2017 < 15.? uses BundlerMinifer NuGet package by default
  - Intended as a stop gap solution to replace gulp/npm
- WebOptimizer is the go forward solution
  - https://github.com/ligershark/WebOptimizer
  - Does more than just bundle/minify
  - Another great tool by Mads Kristensen

ASP.NET CORE WEB OPTIMIZER

- "ASP.NET Core middleware for bundling and minification of CSS and JavaScript files at runtime. With full server-side and client-side caching to ensure high performance. No complicated build process and no hassle."
- Installation:
  - Add package LigerShark.WebOptimizer.Core
  - Update _ViewStart.cshtml
    - @addTagHelper *, WebOptimizer.Core
  - Add app.UseWebOptimizer() to the Configure method
    - Must be called before app.UseStaticFiles()
  - Add services.AddWebOptimizer() to the ConfigureServices method

MINIFICATION

- "Minification is the process of removing all unnecessary characters from source code without changing its functionality in order to make it as small as possible."
- Can minify CSS and JS files
  - Globally or specific files
- Minified files aren't written to disk but cached

BUNDLING

- "Bundling is the process of taking multiple source files and combining them into a single output file. All CSS and JavaScript bundles are also being automatically minified."
- Can bundle CSS and JS files
  - Bundling also minifies files
- Bundles aren't written to disk but cached

VIEW COMPONENTS

VIEW COMPONENTS

- View Components combine server side code with partial views
  - Used to render a chunk of the response
  - Used instead of ChildActions/PartialViews
- Common Uses:
  - Dynamically created menus
  - Login panel
  - Shopping cart

LIMITATIONS

- Can't serve as a client-side end point
- Don't use model binding
- Don't participate in controller lifecycle

CREATE VIEWCOMPONENT CLASS

- Create a new public, non-nested, non-sealed class
- Derive from ViewComponent
  - Can also use name ending in ViewComponent or decorate with [ViewComponent]
- Implement InvokeAsync and return IViewComponentResult
  - Any data needed for view passed into view as viewmodel
  - Typically returns a partial view

CREATE PARTIAL VIEW FOR VIEWCOMPONENT

- Create standard partial view
  - Default name is “default.cshtml"
- Must locate partial view in:

```
Views/<controller_name>/Components/<view_component_name>/<view_name>
Views/Shared/Components/<view_component_name>/<view_name>
```


INVOKING VIEW COMPONENTS

- Invoke from a view (or layout):
  - @Component.InvokeAsync("<name>",<anonymous type with parameters>)
- Invoke from a controller action method:
  - return ViewComponent("<name>“, <anonymous type with parameters>);

- Added in 1.1:
  - Can be invoked as a tag helper from view (or layout)

TAG HELPERS

TAG HELPERS

- Enable server-side code to participate in rendering HTML elements in Razor views
- Reduces the transition between code and markup
  - Tag Helpers Attach to HTML elements
  - HTML Helpers are invoked as methods
- Fully supported with IntelliSense

THE FORM TAG HELPER

- Supported tags (must include at least one):
  - asp-area || asp-controller || asp-action
  - asp-antiforgery (true by default)*
  - asp-route-<parametername> (e.g. asp-route-id="1")
  - asp-all-route-data – uses a dictionary for the route data
  - asp-route (for named routes)
- Area, Controller, Action are defaulted to current route
- Equivalent functionality to @Html.BeginForm/EndForm

FORM FIELD TAG HELPERS

- Include Input, TextArea, Select, and Label tags
- Model property is selected with "asp-for"
  - Provides strong typing with model properties
- Generate Id and Name values based on model property
- Add HTML5 validation attributes based on model definition

FORM FIELD TAG HELPERS (CONTINUED)

- Input (@Html.TextBoxFor or @Html.EditorFor)
  - Adds HTML type based on .NET data type or data annotation
    - E.g. Bool => type="checkbox", [EmailAddress] => type="email"
- Label (@Html.LabelFor)
  - Generates label caption and "for" attribute
- Select (@Html.DropDownListFor or @Html.ListBoxFor)
  - Generates option elements based on "asp-items" attribute

VALIDATION TAG HELPERS

- Validation Message (@Html.ValidationMessageFor)
  - Property selected with asp-validation-for
  - Generates data-valmsg-for attribute
- Validation Summary (@Html.ValidationSummary)
  - Enabled with asp-validation-summary

NON-FORM TAG HELPERS

- Anchor
- Environment
- Link/Script/Image
- Partial
- Cache/Distributed Cache

THE ANCHOR TAG HELPER

- Supported tags:
  - asp-area || asp-controller || asp-action
  - asp-protocol (for http/https)
  - asp-route-<parametername> (e.g. asp-route-id="1")
  - asp-all-route-data – uses a dictionary for the route data
  - asp-route (for named routes)
  - asp-fragment
  - asp-hostname

THE ENVIRONMENT TAG HELPER

- Conditionally renders content based on the run-time environment
- The "names" attribute accepts one or more environment names
  - If HostingEnvironment.EnvironmentName matches, content is loaded
- Changed in 2.0:
  - Added the "include" and "exclude" attributes

THE LINK TAG HELPER

- The "asp-append-version" tag adds hash of file to URL
  - Resolves issue of files still cached when contents change
  - Adds ?v=<hash of file> to the url
- Href handling tags:
  - asp-href-include/exclude – globbed file list to include/exclude
  - asp-fallback-test-class – class used to test original source
  - asp-fallback-test-property – property used to test original source
  - asp-fallback-test-value – value used to test original source
  - asp-fallback-href – fall back file to use if primary file fails to load
  - asp-fallback-href-include || exclude – globbed file list to include/exclude in fall back

THE SCRIPT TAG HELPER

- The "asp-append-version" tag adds hash of file to URL
  - Resolves issue of files still cached when contents change
  - Adds ?v=<hash of file> to the url
- Href handling tags:
  - asp-src-include/exclude – globbed file list to include/exclude
  - asp-fallback-test – script method to test success of loading source
  - asp-fallback-src – fall back file to use if primary file fails to load
  - asp-fallback-src-include || exclude – globbed file list to include/exclude in fall back

THE IMAGE TAG HELPER

- The "asp-append-version" tag adds hash of file to URL
  - Resolves issue of files still cached when contents change
  - Adds ?v=<hash of file> to the url

THE PARTIAL TAG HELPER

- Used to invoke partial views asynchronously
  - Name attribute is required
  - For is the model passed into the view
  - Model used to create a new model instance
  - View-data passes a view data dictionary to the partial
- Used in place of:
  - @await Html.PartialAsync, @await Html.RenderPartialAsync
  - @Html.Partial, @Html.RenderPartial

THE CACHE TAG HELPER

- Provides a way to mark content as cached using the <cache> tag
- Supports absolute, timespan or sliding expiration
- Supports additional cache options:
  - vary-by-header – Cache is evicted when single or list of header values change
  - vary-by-query – Cache is evicted when single or list of query values change
  - vary-by-route – Cache is evicted when single or list of route parameters change
  - vary-by-cookie – Cache is evicted when single or list of cookies change
  - vary-by-user – Cache is evicted when context principal changes
  - vary-by – Allows for further customization of cache data and timeout

THE DISTRIBUTED CACHE TAG HELPER

- Inherits from Cache tag helper
  - All attributes for Cache tag helper are supported
- Supports SQL Server or Redis as a distributed cache

CUSTOM TAG HELPERS

- Composed entirely of server side code
- Class inherits TagHelper
- Class name (minus TagHelper) becomes the element name
  - E.g. EmailTagHelper == <email><email/>
- Public properties are added as lower kebob cased attributes
  - E.g. EmailName == email-name=“”
- Must opt in to use (usually in the _ViewImports.cshtml partial)
  - @addTagHelper *, SpyStore_HOL.MVC

Contact Me

skimedic@outlook.com
www.skimedic.com/blog
www.twitter.com/skimedic

http://bit.ly/skimediclyndacourses
http://bit.ly/apressbooks

www.hallwayconversations.com

Thank You!

Find the code at: https://github.com/skimedic/presentations/tree/master/DOTNETCORE/ASP.NET%20Core/2.1