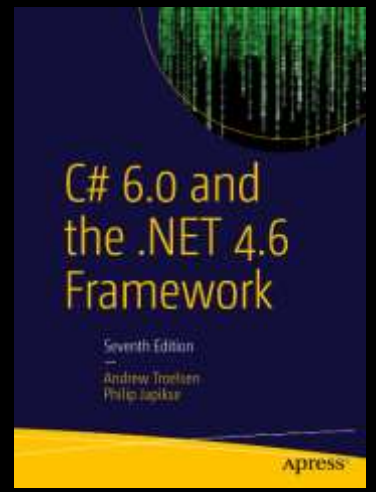
BUILD AN ASP.NET CORE AND EF CORE HANDS ON LAB

Philip Japikse (@skimedic)
skimedic@outlook.com
www.skimedic.com/blog
Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
Consultant, Teacher, Writer

Phil.About()

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



PREREQUISITES

- Visual Studio 2017 (any edition)
- SQL Server 2016 (any edition)
- Lab files from GitHub repo:
  - https://github.com/skimedic/dotnetcore_hol

INTRO TO .NET CORE

WHAT IS .NET CORE?

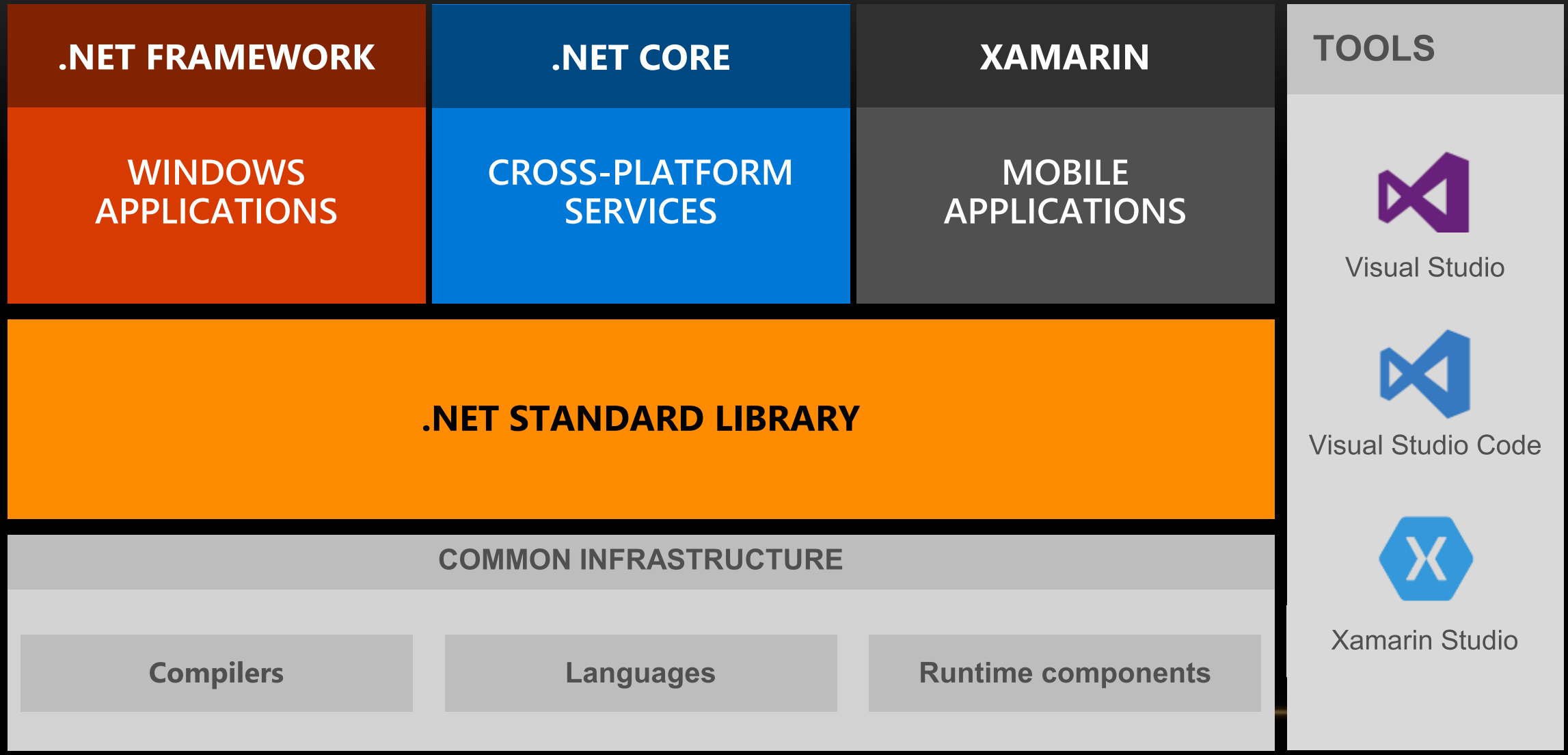
- Rewrite of “full” .NET Framework
- Vast performance improvements over prior versions
  - Including native compilation
- Flexible deployment model
  - Windows, Linux, Mac

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

COMPOSABLE SYSTEM OF NUGET PACKAGES

- Runtime (CoreCLR) –
  - Garbage collection, JIT compiler, base .NET types, low level libraries
- Foundational Libraries (CoreFX) –
  - Collections, file systems, console, XML, async, etc.
- Command Line Interferace (CLI)
- Language Compilers
- Entity Framework Core
- ASP.NET Core

FULL BCD (BIRTHDAY CAKE DIAGRAM)



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

.NET CORE SUPPORT LIFECYCLES

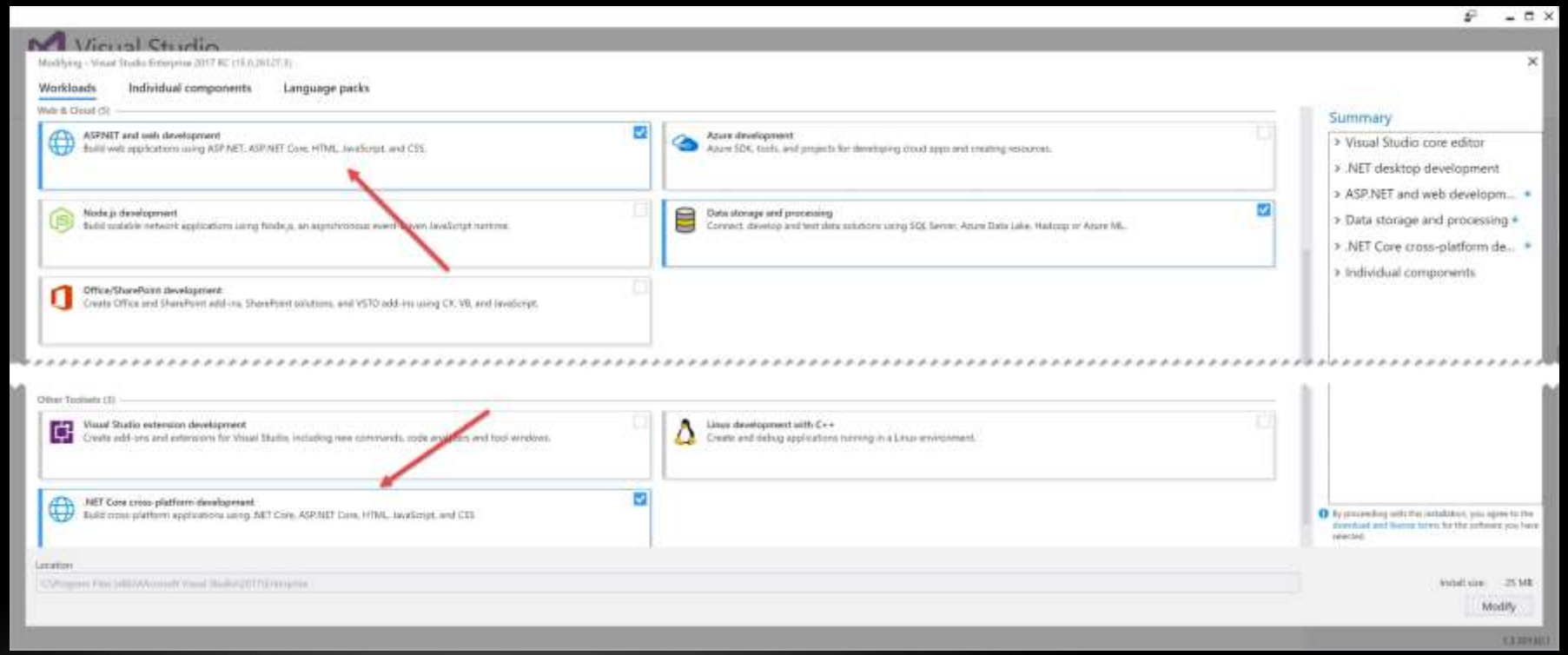
- Long Term Support (LTS)
  - Major releases (e.g. **1**.0, **2**.0)
  - Only upgraded with critical fixes (patches)
  - Supported for three years after GA release or at least one year after the next LTS release.
- Current
  - Minor releases (e.g. 1.**1**, 1.**2**)
  - Upgraded more rapidly
  - Supported for three months after next Current release

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

INSTALLING VS2017 AND .NET CORE

INSTALL VISUAL STUDIO 2017

- Installation process is divided into Workloads
  - Select “ASP.NET and web development” and “.NET Core cross-platform development”



CONFIRM THE INSTALL OF .NET CORE SDK

- Open Command Prompt
  - “where dotnet” => Installations for .NET Core
- Results for the following depend on the path
  - “dotnet” => Shared Framework Host (1.1.0)
  - “dotnet --info” => .NET Core CLI Info (1.0.3)
  - “dotnet --version” => .NET Core CLI Version Number (1.0.3)
  - “dotnet –help” => Lists base CLI commands available

Lab 0: Installing Prereqs

.NET CORE APPLICATIONS ADDING/UPDATING PACKAGES

ANATOMY OF A .NET CORE APPLICATION

- Console Apps and Class libraries are same as under full .NET F/W
- ASP.NET Core apps are simply console applications
  - Create a Web Server on application entry point (Kestrel or IIS)
    - Kestrel – built in web server based on libuv
- Deployment models
  - Self contained – contains everything needed to run, including .NET Core
  - Standard – expects .NET Core installed on deployment machine
- Can execute all with “dotnet run”

PROJECT CONSIDERATIONS

- MVC:
  - Select ‘ASP.NET Core Web Application (.NET Core)’
  - Choose ‘ASP.NET Core 1.1’ Templates
  - Select ‘Web Application’
- Data Access Library
  - Pick ‘Console App (.NET Core)’
  - EF Core migrations require an entry point*
- Models
  - Pick ‘Class Library (.NET Core)’

ADDING/UPDATING NUGET PACKAGES

- NuGet Packages update faster than VS2017 Templates
- Add/Update/Remove packages using:
  - .NET Core Command Line Interface
  - Package Manager Console
  - NuGet Package Manager GUI

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

Lab 1:

Creating the Projects

Adding/Updating the NuGet packages

ENTITY FRAMEWORK CORE

EF PROJECT STATUS

WHAT IS ENTITY FRAMEWORK CORE 1

- Newest version of Entity Framework - complete re-write from EF 6.x
- Lightweight, Modularized
- Cross Platform (built on .NET Core)
- Based on an ‘Opt-in’ model – only load needed packages

- Just released as RTM (1.1.1)
  - Still some missing features from EF 6.x
  - Check http://bit.ly/ef6_efcore to see the current status

(SOME) MISSING* FEATURES IN CURRENT VERSION OF EF CORE 1

- EDMX Designer
  - Not coming back!
- Alternate inheritance mapping patterns
  - Implemented: Table Per Hierarchy (TPH)
  - Missing: Table Per Type (TPT), Table Per Concrete Type (TPC)
- Complex/Value types
- Spatial Data Types
- Lazy loading
- Command Interception
- Stored Procedure Mapping
- Data Initializers
- Some Data Annotations

http://bit.ly/ef6_efcore

EF CORE GOODNESS

DBCONTEXT

- EF Core DbContext changed since EF 6.x
  - Fully embraces dependency injection
- OnConfiguring provides fall back mechanism
- Full support for FluentAPI in OnModelCreating

PERFORMANCE IMPROVEMENTS (NEW)

- EF Core batches multiple insert, update, delete statements into a single call
  - Uses table valued parameters to process changes in a single network call
  - Improved performance through reduced network traffic
  - Reduces cost for cloud based databases
- Batch size can be configured through the DbContextOptions

CONCURRENCY CHECKING (CARRY OVER)

- SQL Server uses Timestamp (rowversion) properties
  - Coded as a byte[] in C#
- Updates and Deletes are modified
  - Where <pk> = @p1 and <timestamp> = @p2
- Error throws DbUpdateConcurrencyException
  - Provides access to entities not updated/deleted
  - EF Core 1.1 added back familiar API calls
- Developer decides how to handle concurrency errors

EF CORE MIGRATIONS (IMPROVED)

- Used to modify schema of based on model and SQL Code
  - Can also scaffold existing database into Context and Models
- Supports more than one DbContext in a project
  - E.g. ApplicationDbContext (ASP.NET Identity) and MyDomainModelContext
- Can also create SQL script representing changes to the database

- Note: Migrations only work with projects that emit entry point

CHANGES FROM EF6 MIGRATIONS

- The Good
  - No longer uses a hash to check database state
  - ModelSnapshot is C# file that contains all of the DDL
  - Database.Migrate method creates model AND runs all migrations

- The bad?
  - Database Initializers and Configuration Seed method are gone

DBSET<T> FIND METHOD (RE-INTRODUCED IN 1.1)

- Introduced in EF Core 1.1
  - Largely due to the developer community
- Searches on primary key(s)
  - Returns instance from DbChangeTracker is currently tracked
  - Else calls to database

USING COMPUTED COLUMNS IN MODELS (FIXED IN 1.1)

- Same table computed columns supported with EF Core 1.0

```
entity.Property(e => e.LineItemTotal).HasColumnType("money")
  .HasComputedColumnSql("[Quantity]*[UnitCost]");
```

- UDF based computed columns supported with EF Core 1.1

```
entity.Property(e => e.OrderTotal).HasColumnType("money")
  .HasComputedColumnSql("Store.GetOrderTotal([id])");
```

FIELD MAPPING/BACKING FIELDS (NEW)

- Allows EF to read and/or write to fields instead of properties
- Conventions
  - [m]_<camel-cased property name>
  - [m]_<property name>
- Fluent API
  - modelBuilder.Entity<Blog>()
    .Property(b=>b.Url).HasField("_theUrl")
- Used when materializing objects
  - Public getters/setters (if they exist) used at other times
- Can control when the fields are used
  - Field
  - FieldDuringConstruction
  - Property

CONNECTION RESILIENCY (RE-INTRODUCED IN 1.1)

- Built in retry mechanism defined by relational database providers
  - Default – no retry
  - SqlServerRetryingExecutionStrategy
    - Optimized for SQL Server and SQL Azure
- Custom Execution Strategy
  - Specify retry count and max delay

- Throws RetryLimitExceededException
  - Actual exception is inner exception

EF CORE SUPPORTS MIXED EVALUATION (NEW)

- EF Core supports queries being evaluated on the server and the client
  - What executes where is provider specific
- Useful for including C# functions into the LINQ query/project
- Be careful where the client functions are injected
  - Poor usage can crush performance
- Enabled or disabled at the context level

```
optionsBuilder.UseSqlServer(connectionString)
  .ConfigureWarnings(warnings =>
        warnings.Throw(RelationalEventId.QueryClientEvaluationWarning));
```

POPULATING MODELS WITH RAW SQL QUERIES (NEW)

- Models can be populated from raw SQL using FromSql on DbSet<T>
  - Select list names must match the names that properties are mapped to
  - All fields on the model must be returned
- Useful for times when Sprocs or UDFs perform better than LINQ/EF
- Can also populate POCOs that are not tables
  - Must be in the Context as a DbSet<T>
  - Must have a primary key defined
- Can be mixed with LINQ statements

Lab 2 Part 1:
Create the Models and ViewModels
Create the DbContext, Migrations
Add Calculated Column to Model (based on UDF)

FINISHING THE DATA ACCESS LAYER

CREATE THE REPOSITORIES

- DbContext is technically a combination of two patterns:
  - Unit of Work
  - Repository
- Adding Custom Repositories eliminates repetitive code
  - Create IRepo<T> and BaseRepo<T> to handle most scenarios
- Specific Repos (e.g. ProductsRepo) handle special cases

DATA INITIALIZATION

- Largely a manual process - Drop/Create base classes from EF 6.x don't exist
- EnsureDeleted - Drops database
- EnsureCreated - Creates database *based on model*
- Migrate()
  - Creates database and runs all migrations as well
  - Usually placed in ctor for DbContext classes
- No way to set initializer – must call from code (Startup.cs?)

Lab 2 Part 2:

Create the Repositories

Create the Data Initializer

TESTING EF CORE

XUNIT TEST FRAMEWORK

- Excellent .NET testing framework
- Built by the creators of nUnit
- Supports full .NET F/W, .NET Core, and Xamarin
  - Project templates are “in the box”
- Supports multitude of test runners
  - VS2017, R#, TestDriven.NET, TeamCity, MSBuild

XUNIT FUNDAMENTALS

- Fact = Test
- Theory = RowTest
- SetUp and TearDown removed in favor of constructors and IDisposable
- ExpectedException removed (Finally!)
- Full Generics support
- Use of anonymous delegates in Assert.Throws

```
Assert.Throws<ExceptionType>(()=>operation());
```

Lab 3:

Testing EF Core

ASP.NET CORE FUNDAMENTALS

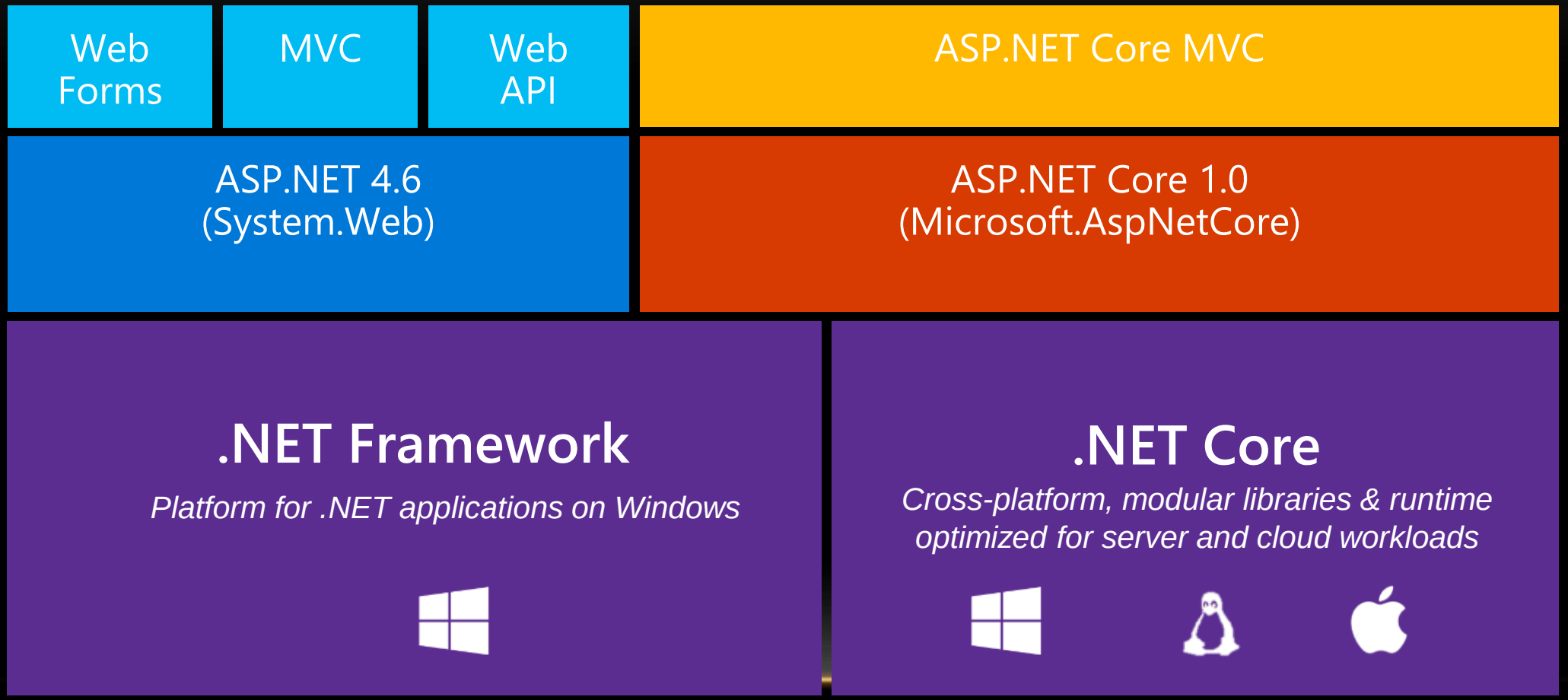
ASP.NET CORE

- ASP.NET Core builds on top of .NET Core
- Single, cross-platform framework for web, services, and microservices
- Fully integrates with CLI tooling and the shared framework
- Takes advantage of .NET Core performance and includes a high performance web server (Kestrel) built on LibUV
- One Framework – WebApi + MVC + Web Pages = ASP.NET Core
- Runs on IIS or Self-Hosted

ASP.NET CORE FEATURES

- Pluggable Middleware enabling you to inject as little or much functionality as needed
  - Routing, authentication, static files, diagnostics, error handling, session, CORS, localization, custom
- Deep integration with Dependency Injection
- Simplified Configuration System
- ***NEW*** Tag Helpers
- ***NEW*** View Components
- Much improved separation of code and content

ASP.NET CORE IN A NUTSHELL



GETTING STARTED WITH ASP.NET CORE

BUILDING THE WEB HOST

- Web Host is configured in the application entry point
- Starts with no services
  - Add in the needed options (typically IIS and Kestrel)
- Defines content root (defaults to wwwroot)
- Defines startup class used to configure the application

STARTUP.CS

- Defined using the WebHostBuilder
- Configures the application through appsettings.json
- Configures services used by the application
  - Populates the Dependency Injection Container
- Configures the HTTP Pipeline

THE STARTUP CLASS

- Constructor loads the application configuration
- Configures the HTTP Pipeline
- Creates services such as MVC, EF, and/or Identity
- Configures the Dependency Injection container

DEPENDENCY INJECTION

- Key component of ASP.NET Core
  - IServiceProvider is built in DI container
  - All aspects of ASP.NET Core can leverage the DI container
- Registering custom interfaces
  - Transient – instantiated separately for every object that needs it
  - Scoped – instantiated once per request
  - Singleton – doesn't need to implement singleton pattern

ADDING CUSTOM SERVICES TO DI CONTAINER

- Any custom service can be added to the DI container
- Built in DI container uses constructor injection
  - Only uses public constructors
  - Must only have one applicable constructor
  - Any additional parameters must have default values
- Services should implement IDisposable
  - Any services created by the container will be automatically disposed

ENVIRONMENTS

- ASP.NET Core references the ASPNETCORE_ENVIRONMENT variable to determine runtime environment
  - Default options are Development, Staging, and Production
  - Custom values are also available
- Greatly simplifies deploying down the operational chain
- Environment can be referenced in code and markup (through tag helpers)

CONFIGURATION

- ASP.NET Core greatly simplifies application configuration
  - Default configuration files are simple JSON files
  - Additional options include command line arguments, in-memory .NET objects, and custom providers
- IConfigurationRoot created by the ConfigurationBuilder class populates the configuration values
  - Typically done in the constructor of the Startup class and added to the DI container
- Can leverages the environment settings

Lab 4:
Creating the WebHost
Configuring the application
Adding connection strings to the settings files

CUSTOM VALIDATION

VALIDATION

- Nothing new for standard model validation
  - Validation and Display attributes
  - Model State
  - Explicit and Implicit validation
- Creating custom validation attributes changed slightly
  - Server side code derives from ValidationAttribute
    - Must also implement IClientModelValidator to support client side scripts
  - Client side validation ties into Jquery validations

SERVER SIDE VALIDATION

- Override ValidationResult IsValid method
  - ValidationContext provides access to metadata and the rest of the model
  - Return ValidationResult.Success or ValidationResult(errorMessage)
- Should also override FormatErrorMessage

CLIENT SIDE VALIDATION

- Must implement AddValidation method in custom attribute
  - Adds the data-val attributes to the rendered element only if using razor editor templates
- JavaScript code needs to:
  - Add validator method – must match data-val name
  - Add unobtrusive validation adapter
    - Must match data-val name
    - Rules must be set to enable validation
- Jquery.Unbotrusive-ajax.js msit be referenced on the page

BUNDLING AND MINIFICATION

BUNDLING AND MINIFICATION

- JavaScript and CSS files should be bundled and minified for performance
- VS 2017 uses BundlerMinifer NuGet package by default
- Settings defined in by bundleconfig.json
  - Specify: outputfilename, inputfiles (globbing allowed), optional parameters
- Add:
  - BundlerMinifier Visual Studio Extension for IDE integration
  - BundlerMinifier.Core for .NET Core CLI

Lab 5:

Custom server and client side validation

Bundling and Minification

VIEW COMPONENTS

VIEW COMPONENTS

- Combine partial views with server side capabilities
- Server side class implements ViewComponent
- Partial view must be located in:
  - Views/<controller_name>/Components/<view_component_name>/<view_name>
  - Views/Shared/Components/<view_component_name>/<view_name>
- Don’t use model binding
- Can be invoked as a Tag Helper (with ASP.NET Core 1.1)

Lab 6:

View Components

VIEW MODELS AND CONTROLLERS

CONTROLLERS

- Everything derives from Controller base class
  - Base class provides many helpers, such as:
  - Ok (200), BadRequest (400), NotFound (404)
- Actions return an IActionResult/Task<IActionResult>
- Dependencies are injected into the controllers
- Attribute Routing is now a first class citizen in ASP.NET Core
  - Helps to refine routing for individual controller actions
- In this example app, base controller OnActionExecuting override is used to create fake authentication

Lab 7:

View Models and Controllers

TAG HELPERS AND VIEWS

TAG HELPERS

- Encapsulate server side code to shape the attached element
  - Keep developers “in the HTML”
- Most Razor HTML Helpers have corresponding Tag Helpers
  - Form, Anchor, Input, TextArea, Select, Validation, Link/Script, Image
  - Added as attributes with asp-
- Special: Environment Tag Helper
- Custom Tag Helpers can be created

TAG HELPER DETAILS

- Form
  - Similar to BeginForm/EndForm HTML Helper
  - Automatically generates the anti-forgery token
  - asp-controller, asp-action, asp-method, asp-route-<parameter name>
  - Can use named routes: asp-route="routename"
- Anchor
  - Similar to ActionLink HTML Helper
  - asp-controller, asp-action, asp-route-<parameter name>

TAG HELPER DETAILS (CONTINUED)

- Input
  - Model property is selected with asp-for (strongly typed)
  - Generates id and name properties for each element
  - Renders markup based on data type of the model property
  - Adds in HTML type based on model type and data annotations
  - Generates HTML5 validation attributes
- TextArea
  - Model property is selected with asp-for (strongly typed)
  - Generates id and name properties for each element

TAG HELPER DETAILS (CONTINUED)

- Select
  - Generates id and name properties for each element
  - Generates select and options using asp-for and asp-items
- Validation
  - Model validation uses asp-validation-summary (in a div)
    - Options are All, ModelOnly, None
  - Property validation uses asp-validation-for (in a span)
- Link, Image
  - Can append (asp-append-version="true") hash of the file to the URL to prevent caching issues

TAG HELPER DETAILS (CONTINUED)

- Script
  - Provide fallback for scripts (such as CDN sources)
    - Fallback source – asp-fallback-src
    - Test for fallback – asp-fallback-test
- Environment
  - Tag helper to define markup block based on configuration environment

```
<environment names="Staging,Production">
```

Lab 8:

Views

CUSTOM TAG HELPERS

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

Lab 9:

Custom Tag Helpers

Questions?

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!