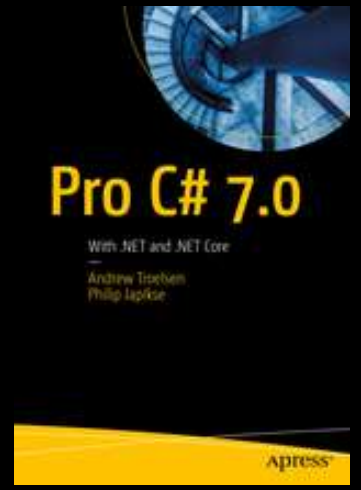
BUILD AN ASP.NET CORE 2.0 AND EF CORE 2.0 APP HANDS ON LAB

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



THE CINCINNATI DAY OF AGILE
CINCY.DEVELOP()

- Http://www.dayofagile.org || http://www.cincydevelop.org – July 27, 2018
- The annual Cincinnati Day of Agile and Cincy.Develop() builds on successful events from past years to present topics on Agile, Delivery, and Development and encourage stimulating conversation for those more advanced in the subject.
- **Sponsors**
  - If you are interested in sponsoring, please contact Phil at admin@dayofagile.org for more information.
- **Mailing List**
  - If you would like to be added to our mailing list, please email us at admin@dayofagile.org with subscribe as the subject.

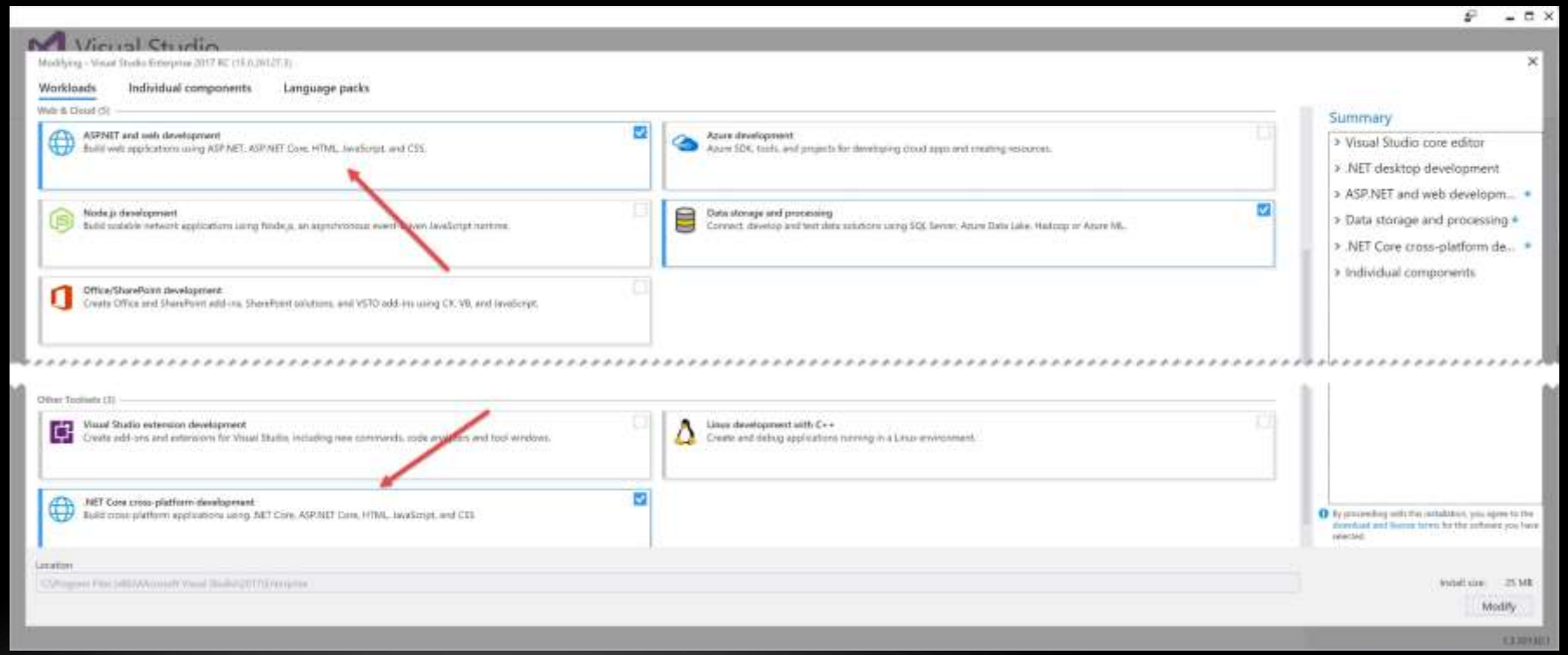
PREREQUISITES

- Visual Studio 2017 15.3+ (any edition)
- .NET Core 2.0 SDK
- SQL Server 2016 (any edition)
- 2.0 Lab files from GitHub repo:
  - https://github.com/skimedic/dotnetcore_hol

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” => “dotnet --info” =>
    - .NET Core CLI Info (2.1.4)
    - Shared Framework Host (2.0.5)
  - “dotnet --version” => .NET Core CLI Version Number (2.1.4)

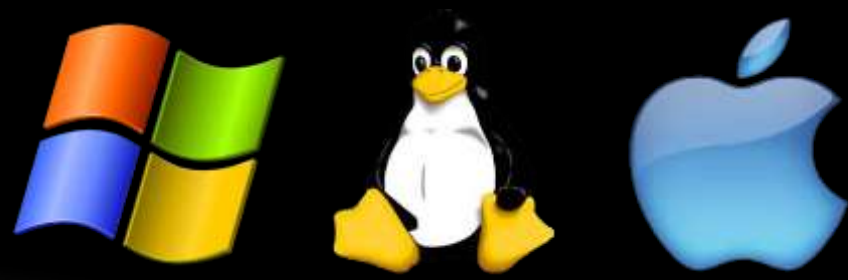
CHANGE THE .NET CORE VERSION FOR A PROJECT

- Create a global.json file at the root

```
{
     "project" : ["src","test"],
     "sdk" : {
          "version" : "1.1.1"
     }
}
```

Lab 0: Installing the Prerequsites

INTRO TO .NET CORE



WHY .NET CORE

- Too many vertical frameworks
  - Desktop/Phone/Silverlight/ASP.NET/Web API
- Microsoft customers and developers asked for:
  - Single unified platform
  - Cross platform support
- Reduce operating costs
- Expand the developer base

NAMING IS HARD

- Original names caused many issues in the .NET community
  - Microsoft didn't want to leave the ASP.NET brand
- The new platforms were rebranded as .NET Core at RTM
  - .NET Core
  - ASP.NET Core
  - Entity Framework Core

COMPOSITION OF .NET CORE

- .NET Core Framework:
  - Runtime (CoreCLR) - GC, jitter, base .NET Types
  - Framework libraries (CoreFX) – Mostly platform agnostic
- Application Host and Command Line Interface (CLI)
- .NET Core Development Frameworks
  - ASP.NET Core
  - Entity Framework Core
- Custom Applications
  - Console applications or class libraries

FULL BCD (BIRTHDAY CAKE DIAGRAM)

.NET FRAMEWORK	.NET CORE	XAMARIN	TOOLS
WINDOWS APPLICATIONS	CROSS-PLATFORM SERVICES	MOBILE APPLICATIONS	 Visual Studio  Visual Studio Code  Xamarin Studio

.NET STANDARD LIBRARY

COMMON INFRASTRUCTURE

Compilers | Languages | Runtime components

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

DEPLOYMENT TARGETS OF ASP.NET CORE

- Windows/Azure using IIS
- Windows/Azure App Service outside of IIS
- On a Linux server using Apache
- On a Linux server using NGINX
- On Windows/Linux using Docker (or other containers)

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
  - Over 32K APIs (from 13K)
  - Also available in Azure Web Apps
- 6 new platforms supported
  - Can target Linux as "single" OS
- .NET Core SDK
  - .NET Core can reference .NET F/W Packages and Projects
  - "dotnet restore" is now implicit
- Performance Improvements
  - Profile-guided optimizations
  - Too many others to list…
- .NET Standard 2.0 NuGet Packages
  - F/W Dependencies removed
- Visual Basic support
  - Console apps, class libraries
- 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

THE PHILOSOPHY OF .NET CORE

CROSS PLATFORM SUPPORT

- Deployable to Windows, Linux, and macOS
  - More and more Linux distributions coming online
- Development on Windows, Linux, and macOS
  - Visual Studio Code
  - Visual Studio for the Mac
- Containerization Support
  - Simplifies deployment and devops
  - Docker support built into the New Project Wizard

PERFORMANCE

- Performance is a first class concern
- Introduction of Kestrel
  - Fast, open source, cross platform web server based on libuv
- ASP.NET performs in top tier of standard benchmark tests
- Additional performance improvements with every release

PORTABLE CLASS LIBRARIES (.NET STANDARD)

- Formal specification for .NET APIs available on all .NET runtimes
- Three key scenarios:
  - Uniform set of base class library APIs, independent of workload
  - Enables production of libraries usable across all .NET implementations
  - Reduces (or eliminates) conditional compilation
- Any assembly targeting a specific .NET Standard version is accessible from any other assembly targeting the same version

PORTABLE OR STAND-ALONE DEPLOYMENT

- .NET Core supports true side by side installation of the platform
  - Installation of additional versions won't affect previously installed versions
- Portable deployments rely on the framework pre-installed
  - Only deploy application specific libraries
- Stand-alone deployments contain required:
  - Application files
  - Framework (CoreFX, CoreCLR) files

MODULAR DEPLOYMENT MODEL

- .NET Core (and related frameworks) are delivered as NuGet packages
- Package names will match assembly names
- Package versions will use semantic versioning
- This provides faster releases of security updates, bug fixes, etc.
- Still supports enterprise with .NET Core distributions

FULL COMMAND LINE SUPPORT

- Command line is a first class citizen
  - Visual Studio tooling typically lags
- Works on all platforms
- Yeoman templates also available

OPEN SOURCE

- Code and documentation are all true open source
- Over 10K contributors for RTM of 1.0
  - Number continues to grow
- Helping to drive adoption in traditionally non-MS organizations

INTEROPERABILITY WITH FULL .NET FRAMEWORK (2.0)

- With .NET Core 2.0, .NET framework libraries can be referenced from .NET Standard libraries.
- Limitations exist:
  - Can only use types supported by .NET Standard*
- .NET Standard 2.0 implements over 32K APIs from the .NET F/W
  - Up from 13K in .NET Standard 1.6

.NET CORE TOOLS TELEMETRY

- Anonymous and aggregated for use by Microsoft and Community Engineers
- Only exists in the tools – not the .NET Core frameworks
- Can disable with the DOTNET_CLI_TELEMETRY_OPTOUT environment variable

SUPPORT LIFECYCLES

.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
- 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

MIGRATION CONSIDERATIONS

WHY MIGRATE EXISTING APPS TO ASP.NET CORE?

- Take advantage of new features
- Increased performance of .NET Core
- Deploy outside of Windows
- .NET Standard 2.0
- See my Migration Courses at http://bit.ly/skimediclyndacourses

WHY *NOT* MIGRATE APPLICATIONS TO ASP.NET CORE?

- What you have works!
- Schedule
- Complexity
- Personnel

BLENDING EF 6 AND EF CORE

- EF Core works with EF 6*
  - Use the same models
  - Context must be in different project
- EF Core brings new features and blazing fast speed

- Bottom line: Start using EF Core NOW

.NET CORE APPLICATIONS ADDING/UPDATING PACKAGES

PROJECT CONSIDERATIONS

- ASP.NET Core Web Applications
  - Select ‘ASP.NET Core Web Application (.NET Core)’
  - Choose ‘ASP.NET Core 2.0’ Templates
  - Select ‘Web Application (Model View Controller)’
- Data Access Library With ASP.NET Core
  - Pick ‘Class Library (.NET Core)’
- Models
  - Pick ‘Class Library (.NET Core)’
- Unit Tests
  - Pick ‘xUnit Test Project (.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 .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

Lab 1:

Creating the Projects

Adding/Updating the NuGet packages

THE CASE FOR OBJECT RELATIONAL MAPPERS (ORMS)

THE PROBLEM

- SQL Databases:
  - Store data relationally using scalar values (with some exceptions)

- Applications:
  - Model data after the domain using complex, object oriented entities

- Translating back and forth takes a lot of code
- Data access code isn't an application differentiator

OBJECT RELATIONAL MAPPING DEFINED

"**Object-relational mapping** (**ORM**, **O/RM**, and **O/R mapping tool**) in computer science is a programming technique for converting data between incompatible type systems using object-oriented programming languages. "

--Wikipedia (https://en.wikipedia.org/wiki/Object-relational_mapping)

ORM BENEFITS

- ORMs convert relational data to domain models and back
  - Removes the need to write database plumbing code
- Developers work with classes that support the domain instead of defined by the confines of SQL based data stores.
  - Focus is on providing business value
- Modern ORMs typically implement:
  - Unit of Work Pattern, Repository Pattern, Implicit Transaction support

UNIT OF WORK DESIGN PATTERN

Maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems.

--Martin Fowler

REPOSITORY DESIGN PATTERN

Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.

--Martin Fowler

IMPLICIT TRANSACTION SUPPORT

- All operations executed in a Unit of Work are wrapped in a transaction
  - Requires ORM and DBMS support
- Explicit transactions can be used for multiple Units of Work

COMMON ARGUMENTS AGAINST ORMS

- Too much “magic”
- “They are slow”
- “Our data is too complex”
- “They are hard to learn”
- The “DBAs won’t let us”

ORM USE CASES

The Bad

- Reporting solutions
- ETL Operations
- Set based operations

The Good

- CRUD operations
- Forms over data
- Line of business applications

IT'S NOT THE RING OF POWER!

ENTITY FRAMEWORK CORE

EF PROJECT STATUS

WHAT IS ENTITY FRAMEWORK CORE

- 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 EF Core 2.0
  - Many more features added
  - 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 2

- EDMX Designer
  - Not coming back!
- Spatial Data Types
- Command Interception
- Alternate inheritance mapping patterns
  - Implemented: Table Per Hierarchy (TPH)
  - Missing: Table Per Type (TPT), Table Per Concrete Type (TPC)
- Stored Procedure Mapping
- Some Data Annotations
- ~~Raw SQL (Non-Entity Types)~~
- ~~Lazy loading~~
- ~~Data Initialization~~
- ~~Group By Translation~~

http://bit.ly/ef6_efcore

DATA ANNOTATIONS REMOVED/CHANGED IN EF CORE

Removed

- Only works in Fluent API
  - Composite Keys
  - Index, Composite Indices
- ComplexType
  - Fluent API (2.0)
  - Owned attribute (2.1)

Changed

- Table
  - Must specify schema properly

FEATURES ADDED TO EF CORE (NOT IN EF 6)

- Batching of Statements (1.0)
- Shadow State Properties (1.0)
- Alternate Keys (1.0)
- Client side key generation (1.0)
- Mixed Client/Server evaluation (1.0)
- Raw SQL with LINQ (1.0)
- Field Mapping (1.1)
- DbContext Pooling (2.0)
- Like query operator (2.0)
- Global Query Filters (2.0)
- String interpolation with raw SQL (2.0)
- Scalar function mapping (2.0)
- Explicitly compiled LINQ queries (2.0)
- Attach a graph of new and existing entities (2.0)

FEATURES ADDED TO EF CORE IN 2.1 PREVIEW 1 (NOT IN EF 6)

- Entity ctors with parameters (2.1)
- Property Value Conversions (2,1)
- Query Types w/o keys (2.1)
- Eager loading for derived types (2.1)

- NOTE: More to come, with monthly releases of 2.1

WHEN CHOOSING EF 6.X VS EF CORE 1, CHOSE WISELY

EF 6.x

- Windows only
- Full featured
  - 8 years of development
  - Patches and minor releases still coming
- Widely used
  - Rich set of database providers
  - Lots of articles, books, examples

EF Core

- Windows or Cross Platform
- Brand new - missing features from EF 6
  - EF Core 2.0 greatly closed the gap
  - Documentation is catching up
- Many Improvements over EF 6.x
  - Much faster performance
  - New features not in EF 6.x
- Works with EF 6.x*

EF COMPONENTS

MAJOR COMPONENTS OF EF

- DbContext
- ChangeTracker
- DbSet
- Entities
- Database Providers

DBCONTEXT

- Represents a session with the database and is used in query and save operations
  - Database property exposes a DatabaseFacade for db related info and operations
- Implements a combination of the Unit of Work and Repository patterns
- Provides additional model configuration via Fluent API
- Is configured using DbContextOptions/DbContextOptionsBuilder

DBCONTEXT CHANGES FROM EF 6

- Fully embraces dependency injection
- Requires DbContextOptions for configuration
  - OnConfiguring provides fall back mechanism
  - DbContext Design Time Factory (new)

DBCONTEXT DESIGN TIME FACTORY

- EF Core Tooling Commands requires parameterless ctor or
- IDesignTimeDbContextFactory<TContext>
  - Used by EF Migrations to create DbContext

DBCONTEXT POOLING (2.0)

- Works in conjunction with ASP.NET Core 2
- Creates a pool of DbContext instances
- Resets the Change Tracker between uses

- Only works for one DbContext
  - Last one in wins

CHANGETRACKER

- Provides access to change tracking information and operations for entity instances tracked by the context.
- Also tracks the original values
- Works with the DbContext when SaveChanges is called

DBSET<T> WHERE T: {ENTITY CLASS}

- Represents a collection of all entities of a given type in a Context.
  - Implements IQueryable<T> in addition to collection interfaces
- Can be used to query and save instances of entities.
- LINQ statements used against a DbSet<T> are translated into queries against the data store.
  - Some parts might be evaluated in memory depending on the data store
- Items removed, added, or changed in a DbSet are not persisted until SaveChanges is called on the Context.

ENTITIES

- Simple .NET classes designed to model the domain
  - Often referred to as Plain Old CLR Objects (POCOs)
- Mapped to the relational data through conventions, data annotations and/or the fluent API
- Relate to other entities in the model through navigation properties

EF CORE PROVIDER MODEL

- EF Core uses a provider model to allow use with different data stores
- Concepts common to most providers are contained in the core components
- Provider specific concepts are in specific components

DATABASE PROVIDERS AVAILABLE

Third Party support:

- MySql (Official, Pomelo)
- SQLite
- PostgreSQL (Third party)
- Db2 (IBM)
- Oracle (paid only)
- MyCat
- Firebird
- SQL Compact

Supported by EF Core directly:

- SQL Server
  - Used in this course
- SQLite
- InMemory
  - Covered in Essential EF Core 2.0 Part 2 with testing

EF CORE GOODNESS

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 persisted
- Developer decides how to handle concurrency errors

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 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

PERFORMANCE IMPROVEMENTS (1.0, 1.1, 2.0, 2.1)

- 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

MIXED CLIENT/SERVER EVALUATION (1.0)

- 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 (1.0)

- 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

GLOBAL QUERY FILTERS (2.0)

- Model level filters defined directly on the model
- Automatically applied to any queries on that type
  - Also applied to indirect queries (e.g. using Include or ThenInclude)
  - Can be used for soft deletes or multi-tenancy

```
public class BloggingContext : DbContext
{
    public DbSet<Post> Posts { get; set; }
    public int TenantId {get; set; }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Post>().HasQueryFilter(
            p => !p.IsDeleted && p.TenantId == this.TenantId );
    }
}
```

STRING INTERPOLATION WITH RAW SQL QUERIES (2.0)

- Implemented in FromSql and ExecuteSqlCommand
- C# string interpolation items get converted into SQL parameters

```
var city = "London";
var contactTitle = "Sales Representative";
using (var context = CreateContext())
{
  context.Set<Customer>()
    .FromSql($@"SELECT * FROM ""Customers""
        WHERE ""City"" = {city} AND
           ""ContactTitle"" = {contactTitle}")
    .ToArray();
  }
```

```
@p0='London' (Size = 4000)
@p1='Sales Representative' (Size = 4000)

SELECT *
FROM ""Customers""
WHERE ""City"" = @p0
    AND ""ContactTitle"" = @p1
```

SCALAR FUNCTION MAPPING & EXPLICITLY COMPILED QUERIES (2.0)

- Scalar Function Mapping
  - Can map scalar functions to C# methods and used in LINQ queries
    - Use DbFunctionAttribute and make the method static on DbContext
    - Must create the function yourself
- Explicitly compiled queries
  - Create LINQ queries as static C# functions using EF.CompileQuery
  - Bypasses computation of hash and cache lookup

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() - Mutually exclusive of EnsureCreated()
  - Creates database and runs all migrations as well
- No way to set EF 6 style initializer – must call from code

Lab 2 Parts 1,2,3:

Create the Data Access Layer

ASP.NET CORE 2.0 FUNDAMENTALS

ASP.NET CORE 2.0

- ASP.NET Core 2.0 rebuilt on top of .NET Core 2.0
- 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

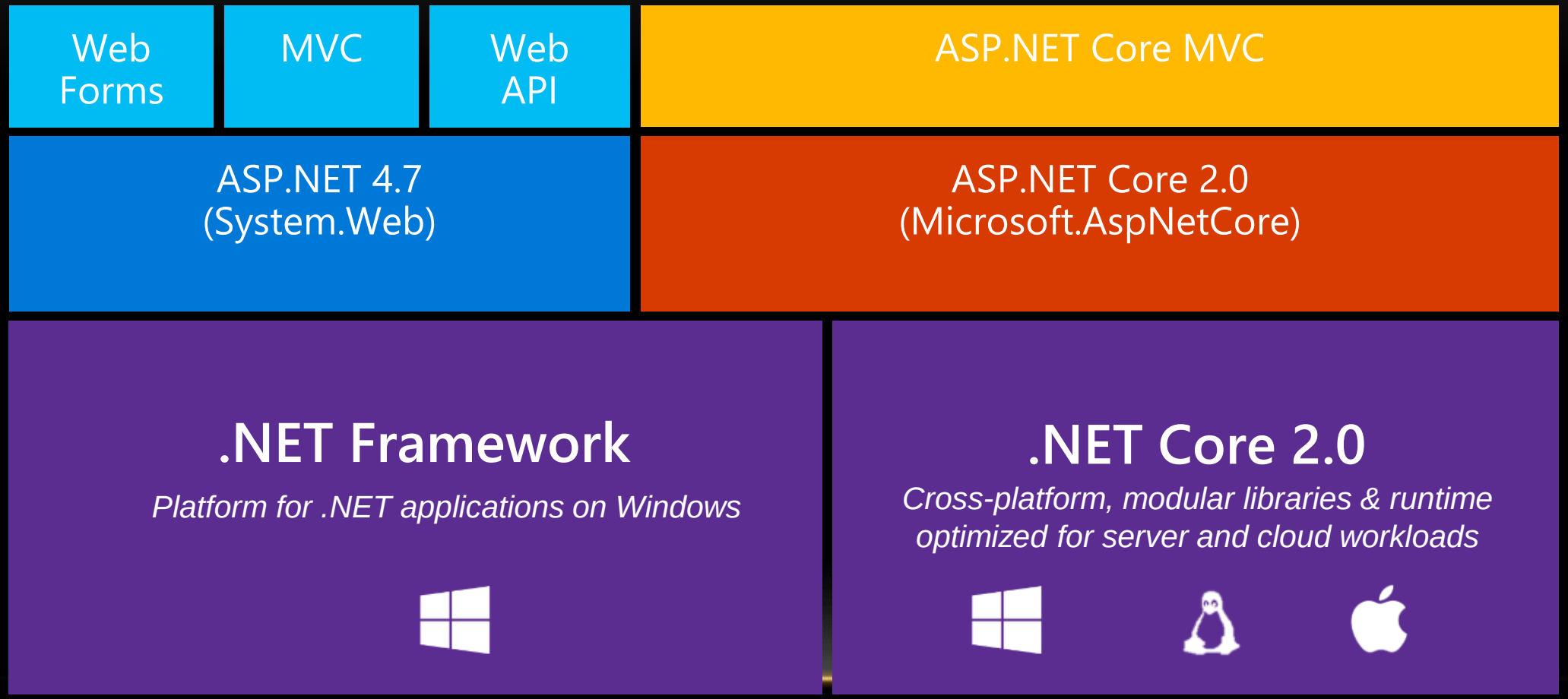
WHAT'S NEW IN ASP.NET CORE 2.0

- 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 PREVIEW

- 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 BCAD



CONFIGURING THE WEB SERVER(S)

CREATE A WEB HOST

- ASP.NET Core applications are console applications that create a web host
  - Kestrel in 1.0
- The Web Host is created with the WebHostBuilder class
  - Responsible for app startup and lifetime management
- At a minimum, it configures a server and request processing pipeline

CONFIGURE THE WEB HOST

- Additional features and configuration are opt-in
  - UseIISIntegration adds IIS with reverse proxy
  - UseStartup<T> adds configuration (Configure and ConfigureServices)
  - Plus many more
- Changed in 2.0:
  - CreateDefaultBuilder configures standard IWebHostBuilder
  - Many additional options added to WebHostBuilder

USING SSL LOCALLY - IIS

- Check Enable SSL on Debug project property page (note the port)
- Configure MVC for SSLPort (44300 or greater)
- Add RequireHttps attribute/filter/etc.
- Note: If you have not trusted your IIS self signed cert, you will get an exception in the browser
- Added in 1.1:
  - URL Rewriting to HTTPS

USING SSL LOCALLY - KESTREL

- With Kestrel:
  - Install the Desktop Development with C++ Workload
  - Need the Microsoft.AspNetCore.Server.Kestrel.Https package
  - Make a cert, convert to pfx format. Refer to CreateCert.txt solution item
  - Make pfx file into X509 certificate, add it to Kestrel options, specify ports
  - Add RequireHttps attribute, configure MVC for SSLPort
- Added in 1.1:
  - URL Rewriting to HTTPS
- Changed in 2.0:
  - Kestrel configuration (breaking change)

LAUNCHSETTINGS.JSON CONTROLS RUNNING APP IF PRESENT

- 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 THE PIPELINE

- The Configure method sets up how to respond to HTTP requests

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
   ILoggerFactory loggerFactory)
{
      app.UseExceptionHandler("/Home/Error");
      app.UseStaticFiles();
      app.UseMvc(routes =>
      {
          routes.MapRoute(
             name: "default",
             template: "{controller=Home}/{action=Index}/{id?}");
      });
}
```

CONDITIONAL PIPELINE CONFIGURATION

- Use environment options for conditional pipeline configuration

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env,
    ILoggerFactory loggerFactory)
{
        if (env.IsDevelopment())
        {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
        }
        else
        {
                app.UseExceptionHandler("/Home/Error");
        }
}
```

CONFIGURING FRAMEWORK SERVICES

- Used to configure any services needed by the application

```
public void ConfigureServices(IServiceCollection services)
{
   // Add framework services.
   services.AddMvc(config =>
   {
      config.Filters.Add(new SimpleAuthenticationActionFilter());
   })
   .AddJsonOptions(options =>
   { //Revert to PascalCasing for JSON handling
      options.SerializerSettings.ContractResolver = new DefaultContractResolver();
   });
   //Additional services for DI added here (covered later in this presentation)
}
```

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

Lab 3:

Creating the WebHost

Configuring the application

Adding connection strings to the settings files

VIEW MODELS AND CONTROLLERS

CONTROLLERS AND ACTIONS

- Controller, AsyncController, APIController all rolled into one
- All return IActionResult (or Task<IActionResult>)
- Many helper methods built into base Controller for returning HttpStatusCodes
  - NoContent (204), OK (200), BadRequest (400), etc.
- API methods must specify HTTP Verb
  - No long based on name of method

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

Lab 4:

View Models and Controllers

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
- 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 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

Lab 5:

Views

MANAGING CLIENT SIDE LIBRARIES

MANAGING CLIENT SIDE LIBRARIES

- Bower is dead
- Library Manager is coming in VS 15.7
  - https://github.com/aspnet/LibraryManager
  - Download, compile, install VSIX
- Libraries are managed in Library.json
  - Cdnjs is default library source
  - Can be configure per package or globally
- Another great tool by Mads Kristensen

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)

Lab 6:

View Components

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 7 [Optional]:

Custom Tag Helpers

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 must be referenced on the page

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

add app.UseWebOptimizer() to the Configure method anywhere before app.UseStaticFiles

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

FILTERS

FILTERS

- *Filters* in ASP.NET MVC allow you to run code before or after a particular stage in the execution pipeline.
  - Filters can be configured globally, per-controller, or per-action.
- Many types available
  - Action
  - Exception
  - Authorization
  - Resource
  - Result

EXCEPTION FILTERS

- Come into play on unhandled exceptions

```
public class SpyStoreExceptionFilter : IExceptionFilter
{
        public void OnException(ExceptionContext context)
        {
                var ex = context.Exception;
                var response = new ErrorMessage {Message = ex.Message };
                context.Result = new ObjectResult(response) {StatusCode = 500};
        }
}
```

- Configured with MVC (for whole application filters)

```
services.AddMvc(config => config
  .Filters.Add(new SpyStoreExceptionFilter(_env.IsDevelopment()))
  .Filters.Add(new SimpleAuthenticationFilter()));
```

ACTION FILTERS

- Come into play before and/or after actions are executed

```
public class SimpleAuthenticationActionFilter : IActionFilter
{
        public void OnActionExecuting(ActionExecutingContext context)
        {   // do something before the action executes
        }
        public void OnActionExecuted(ActionExecutedContext context)
        {  // do something after the action executes
        }
}
```

Lab 8 [Optional]:

Custom server and client side validation

Bundling and Minification

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 9 [Optional]:

Testing EF Core

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!