ASP.NET Core MVC for ASP.NET MVC 5 Developers

Speaker: Mike Smith

Instruction from the Trenches

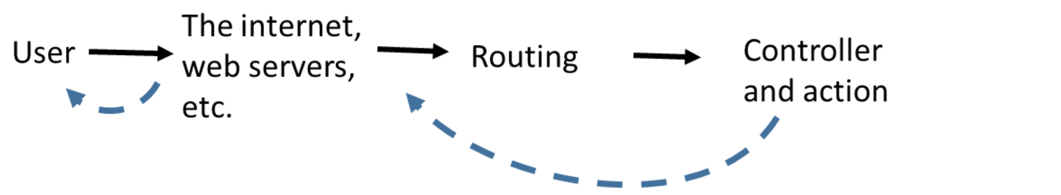
Mike Smith

- MCT, MVP, MCPD, MCAD, MCDBA, MCITP: DBA, …..
- In the microcomputer field since 1980
- Trainer / Developer / Tech Writer
- Specializing in:
  - ✓ SharePoint
  - ✓ SQL
  - ✓ .NET, C#, VB and related .NET technologies
  - ✓ PowerShell
  - ✓ Azure
  - ✓ and lots of stuff from way back when…
- Email address: mike@maxtrain.com
- Blog: TechTrainingNotes.blogspot.com

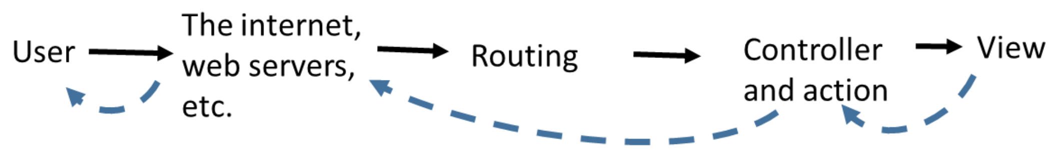
MAX
max technical training

MVC Core

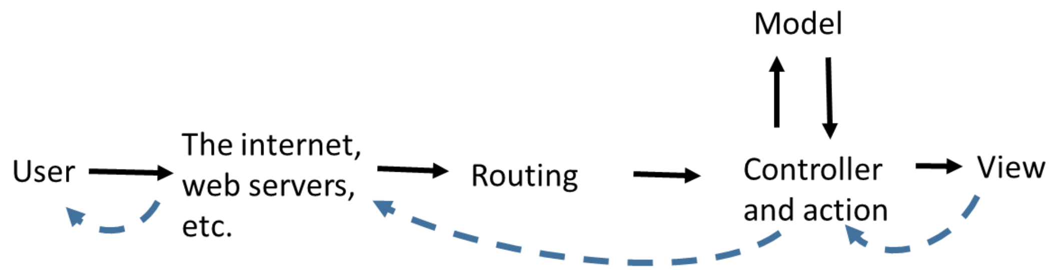
The same…
but different



Action returns text, HTML, JSON or XML.



No data needed from the model. Action returns a View. (which typically is HTML text)



Data retrieved (or set) in the model. Action returns a View (which typically is HTML text) using data from the Model.

- Same design pattern.
- Still Routing > Controller > Model > View

  RCMV, but some people insist in calling it MVC.

Web Project Types

- Web Project Types
  - No VB.Net templates! (But there are for Console apps! ☺)

Full Framework	.NET Core
Empty	Empty
Web API	API
Web Forms (not in Core)	Web Application (Razor Pages) *(new)*
MVC	Web Application (MVC)
Single Page Application (not in Core)	Razor Class Library *(new)*
Azure API App (not in Core)	Angular *(new)*
	React.js *(new)*
	React.js and Redux *(new)*

Web Servers and Hosting

ASP.NET

- Everything is IIS!

ASP.NET Core

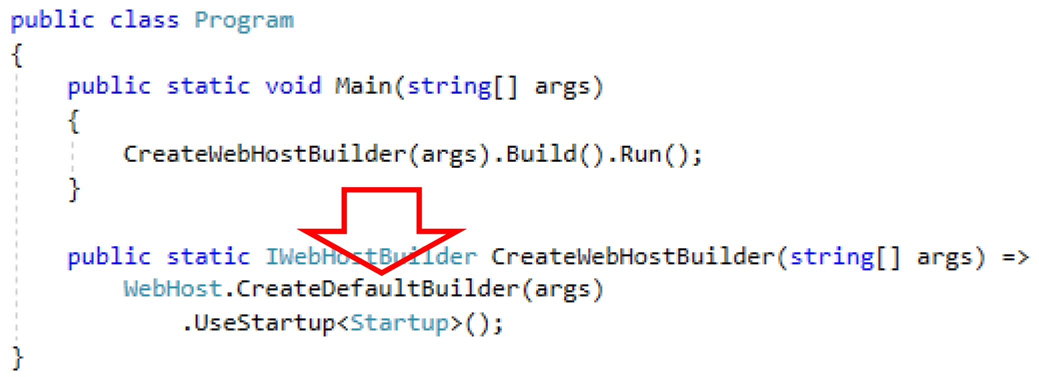
- Kestrel
- Kestrel + reverse proxy (IIS, Nginx, Apache)
- HTTP.sys (Windows servers only)

Kestrel

- A bird of the falcon genus
- An rocket engine. The Kestrel engine was developed in the 2000s by SpaceX for upper stage use on the Falcon 1 rocket.
- An airplane (predecessor to the Harrier).
- A web server.

Kestrel

- A GitHub project available as a Nuget package, and included in the Microsoft.AspNetCore.App metapackage (ASP.NET Core 2.1 or later)..
- Supported on all platforms and versions that .NET Core supports.
- ASP.NET Core project templates use Kestrel by default.
  - Program.cs/Main calls CreateDefaultBuilder which internally calls UseKestrel.



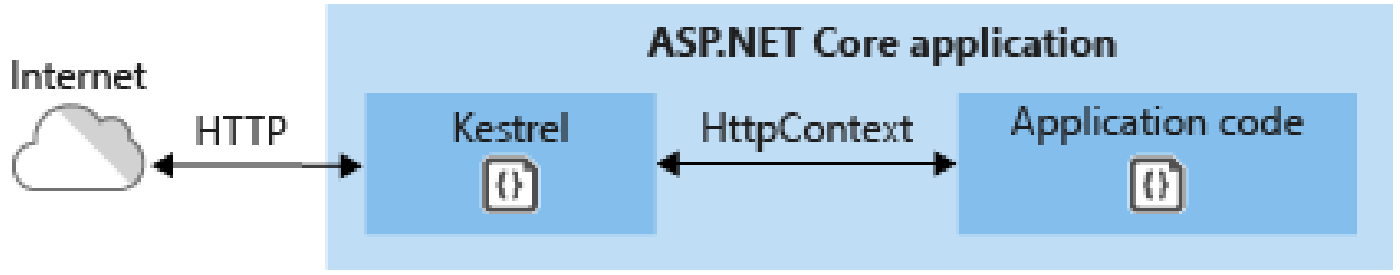
```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

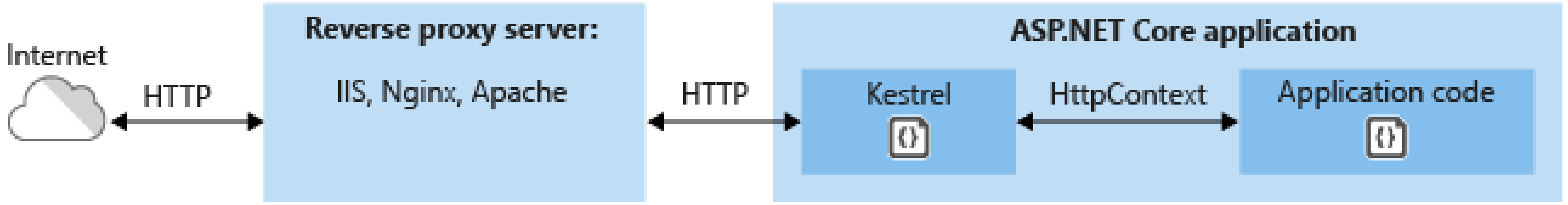
https://github.com/aspnet/AspNetCore/tree/2e7c52d97c1aa9b372d740fd46accfd6e2d5a97b/src/Servers/Kestrel
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-2.2

Kestrel

- As a web server

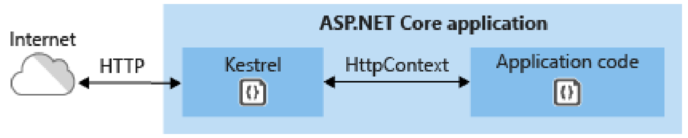


- Behind a Reverse Proxy server

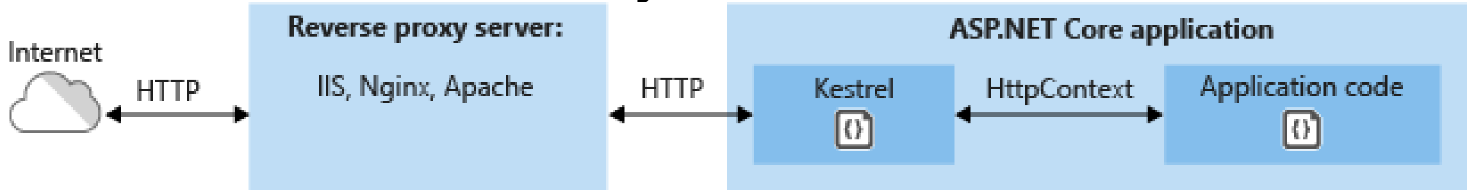


https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-2.2

Kestrel



- Behind a Reverse Proxy server

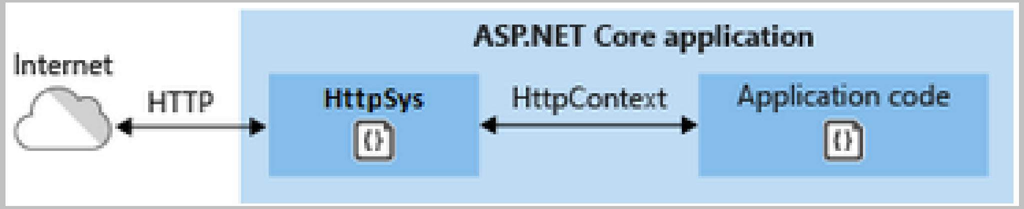


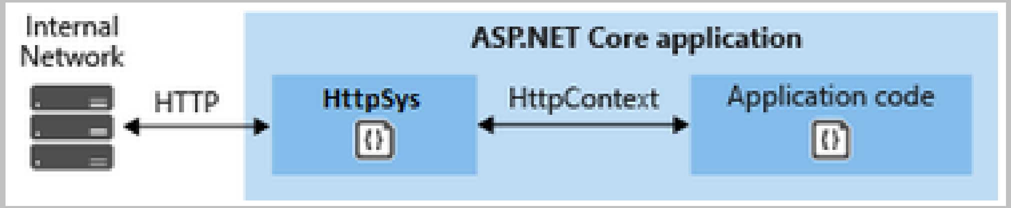
- A reverse proxy:
  - Can limit the exposed public surface area of the apps that it hosts.
  - Provide an additional layer of configuration and defense.
  - Might integrate better with existing infrastructure.
  - Simplify load balancing and secure communication (HTTPS) configuration. Only the reverse proxy server requires an X.509 certificate, and that server can communicate with your app servers on the internal network using plain HTTP.
- Kestrel used as an edge server without a reverse proxy server doesn't support sharing the same IP and port among multiple processes. (Kestrel takes all requests for a port regardless of host headers.)

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-2.2#when-to-use-kestrel-with-a-reverse-proxy

HTTP.sys

- HTTP.sys is a web server for ASP.NET Core that only runs on Windows and offers some features that Kestrel does not.
  - Use when there's a need to expose the server directly to the Internet without using IIS.
  - Use when an internal deployment requires a feature not available in Kestrel, such as Windows Authentication.





View Engines

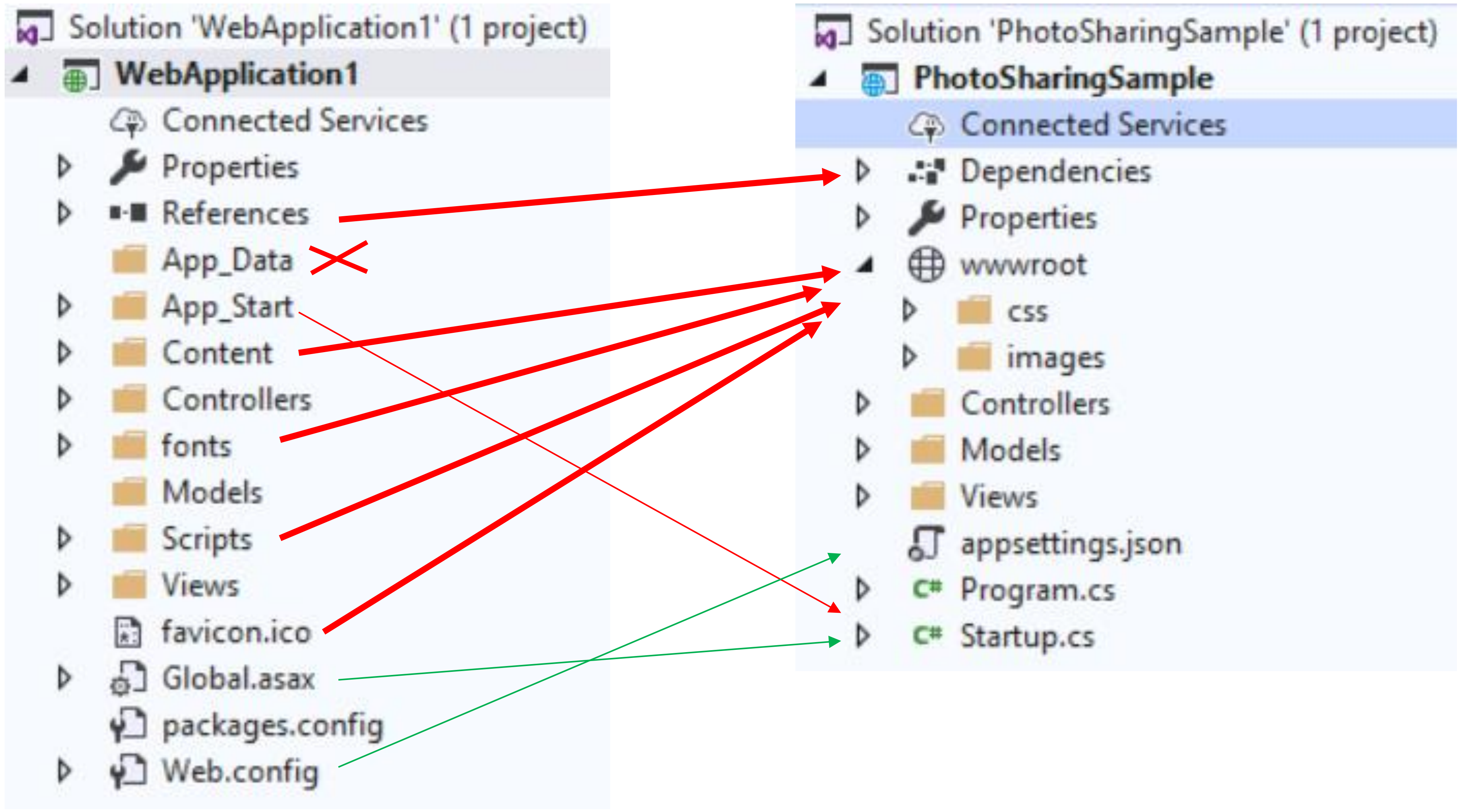
.NET Full Framework

- **Razor** and Web Forms
- And Brail, NDjango, NHaml, NVelocity, SharpTiles, Spark, StringTemplate and XSLT

.NET Core

- **Razor**
- A few demo, but not production, projects: Markdown, NVue and others.
- In Docs:
  In ASP.NET Core MVC, views are .cshtml files that use the C# programming language in Razor markup.

Project Folder Changes



Solution 'WebApplication1' (1 project)
- WebApplication1
  - Connected Services
  - Properties
  - References
  - App_Data
  - App_Start
  - Content
  - Controllers
  - fonts
  - Models
  - Scripts
  - Views
  - favicon.ico
  - Global.asax
  - packages.config
  - Web.config

Solution 'PhotoSharingSample' (1 project)
- PhotoSharingSample
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
    - css
    - images
  - Controllers
  - Models
  - Views
  - appsettings.json
  - Program.cs
  - Startup.cs

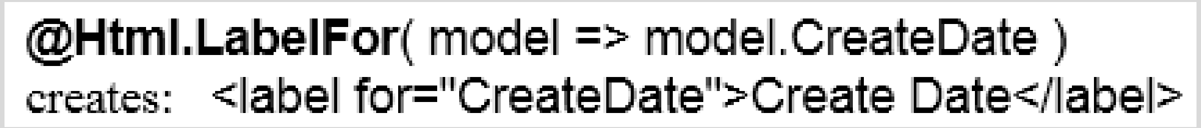
What's New / Changed in Razor Views

- Old code should still work.
- "@" and HTML helpers still work the same.

…

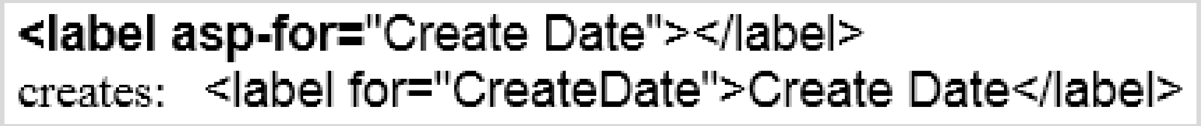
What's New / Changed in Razor Views

- Tag Helpers
  - Razor supplies an HTML class with a collection of methods that create HTML tags.



```
@Html.LabelFor( model => model.CreateDate )
creates:   <label for="CreateDate">Create Date</label>
```

  - .NET Core Razor also adds Tag helpers for similar tasks.
    - Not every Html helper currently has a Tag helper.



```
<label asp-for="Create Date"></label>
creates:   <label for="CreateDate">Create Date</label>
```

Tag Helper	Equivalent HTML helper
Anchor	Html.ActionLink
Cache	n/a
Distributed Cache	n/a
Environment	n/a
Form	Html.BeginForm and Html.BeginRouteForm
Image	An image tag plus a URL helper
Input	Html.TextBoxFor, Html.EditorFor and others.
Label	LabelFor
Partial	Html.Partial and Html.RenderPartial
Select	Html.DropDownListFor and Html.ListBoxFor
Textarea	Html.TextAreaFor
Validation Message Validation Summary	Html.ValidationSummary

Razor Page Projects

- Not the topic of this presentation… but…
- Similar to MVC, but each page has it’s own controller.
  - More like Model-View-ViewModel
  - May be faster and easier to create for smaller projects.
- A page focus instead of a controller focus.
  - Code for the page is “near” the page.
  - Kind of like a controller, but with only Get and Post actions.
- Two files: pagename.cshtml and pagename.cshtml.cs
  - Page with code-behind? Sounds familiar!

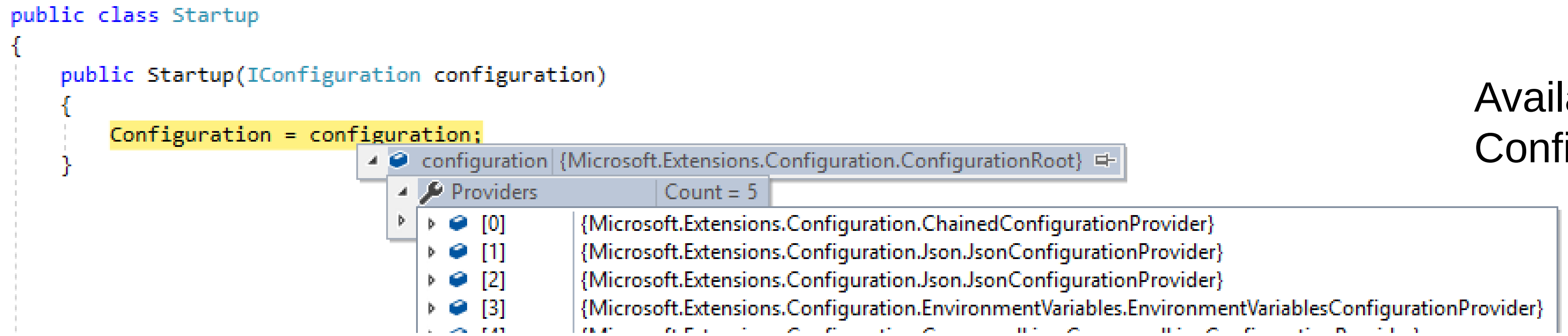
Razor Class Library

- “Razor views, pages, controllers, page models, Razor components, View components, and data models can be built into a Razor class library (RCL). The RCL can be packaged and reused. ”
- Compiles into a DLL.
- New in 2.1

https://docs.microsoft.com/en-us/aspnet/core/razor-pages/ui-class?view=aspnetcore-2.2&tabs=visual-studio

Configuration

- ASP.NET
  - Web.Config
  - Loaded by IIS or read using ConfigurationManager.AppSettings
- ASP.NET Core
  - Store settings anywhere and load using middleware.
  - Default is appsetting.json (and Properties/launchSettings.json)
  - Plus Environment variables.
  - By default, all are passed into the Startup constructor.



```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }
```

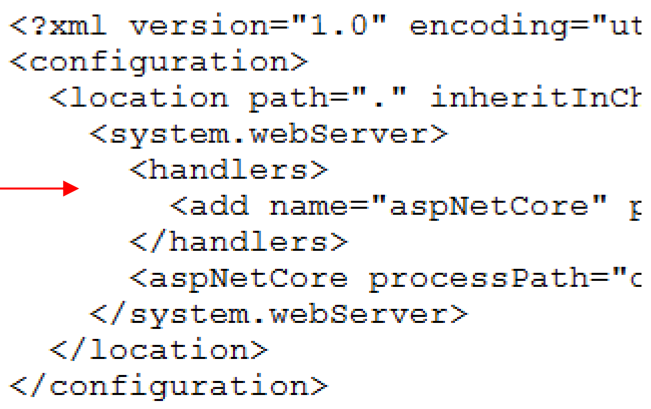
```
configuration {Microsoft.Extensions.Configuration.ConfigurationRoot}
  Providers   Count = 5
    [0]   {Microsoft.Extensions.Configuration.ChainedConfigurationProvider}
    [1]   {Microsoft.Extensions.Configuration.Json.JsonConfigurationProvider}
    [2]   {Microsoft.Extensions.Configuration.Json.JsonConfigurationProvider}
    [3]   {Microsoft.Extensions.Configuration.EnvironmentVariables.EnvironmentVariablesConfigurationProvider}
```

Available from the Configuration object.

Configuration

- ASP.NET Framework
  - Web.config (and parent web.config and machine.config)
  - Or anyplace else you like using your own code: XML, SQL, etc.
  - Web.config loaded by IIS, ASP.NET and other tools.

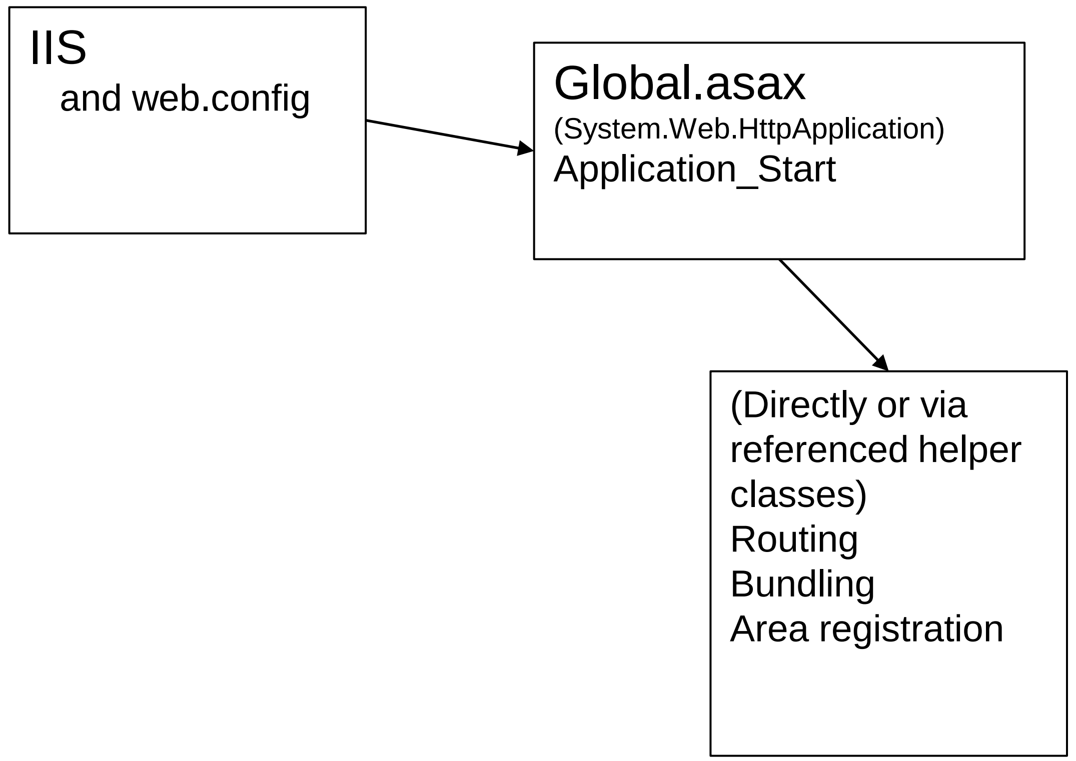
- ASP.NET Core
  - appsettings.json (Loaded by CreateWebHostBuilder in Program.cs)
  - appsettings.*{Environment}*.json `appsettings.Release.json`
  - Web.config (maybe, but just to route to the aspNetCore handler)
    - Auto-generated when publishing to IIS.
    - Can be added to a project when custom IIS settings are needed.
      The "auto-generated" content is added to the file during a publish to IIS.
  - Environment variables (Windows or Linux) named ASPNETCORE_*???*
    - This lets you store development vs production settings outside of the project.
      `if (env.IsStaging() || env.IsEnvironment("Staging_2"))`
  - Or anyplace else you like: using your own code or using a Configuration Provider
    - JSON, XML or INI *- remember INI files?*



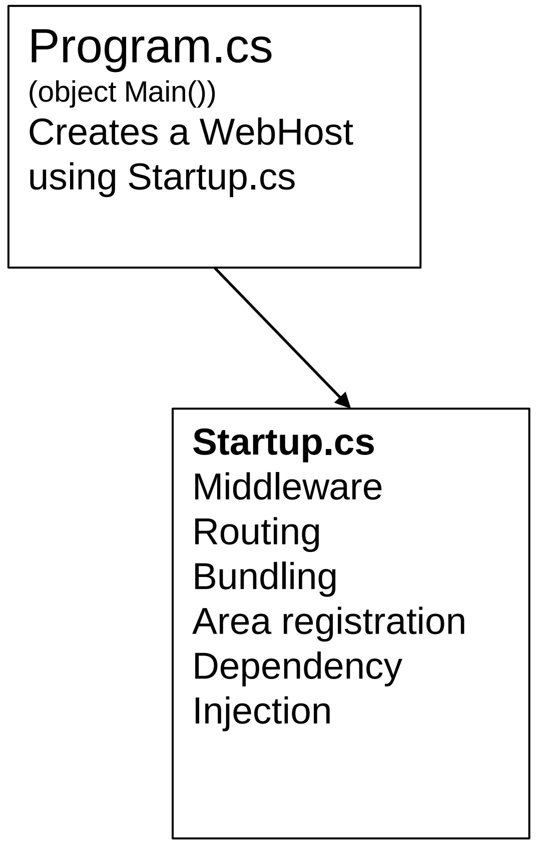
```
<?xml version="1.0" encoding="ut
<configuration>
  <location path="." inheritInCh
    <system.webServer>
      <handlers>
        <add name="aspNetCore" p
      </handlers>
      <aspNetCore processPath="c
    </system.webServer>
  </location>
</configuration>
```

https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/environments

Full FrameworK

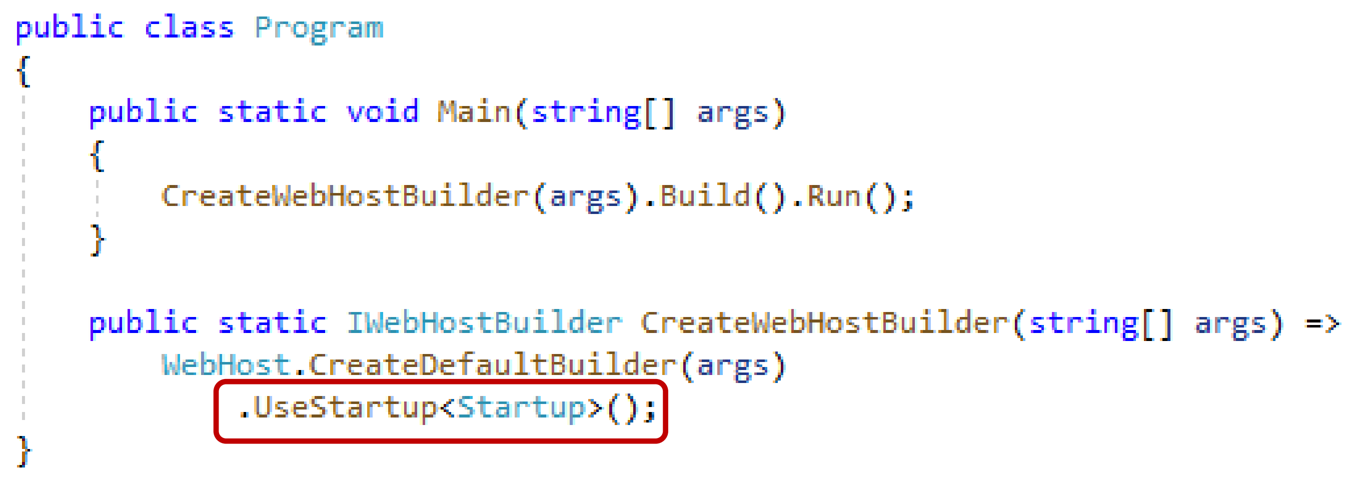


Core



Program.cs

- Initial Entry Point.
- Creates the hosting environment.
- Passes the Startup class to host.

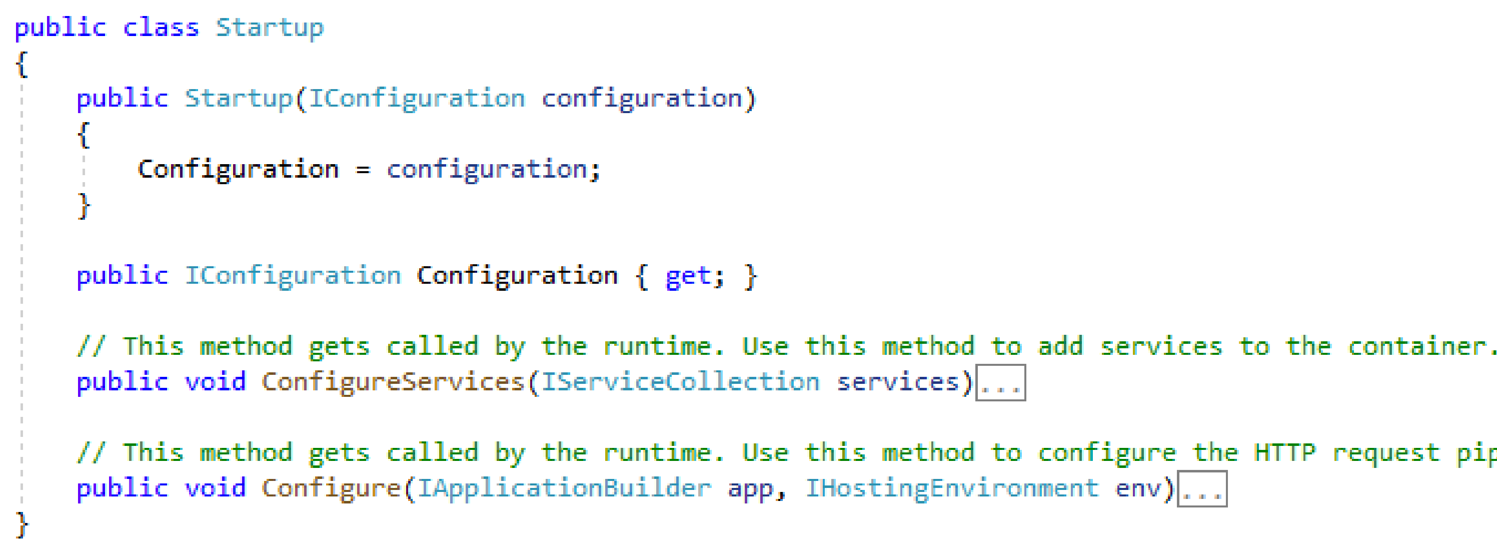


```
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup<Startup>();
}
```

Startup.cs

- Contains:
  - A constructor that receives the configuration object. (a set of key/value application configuration properties)
  - A ConfigureServices method
  - A Configure method



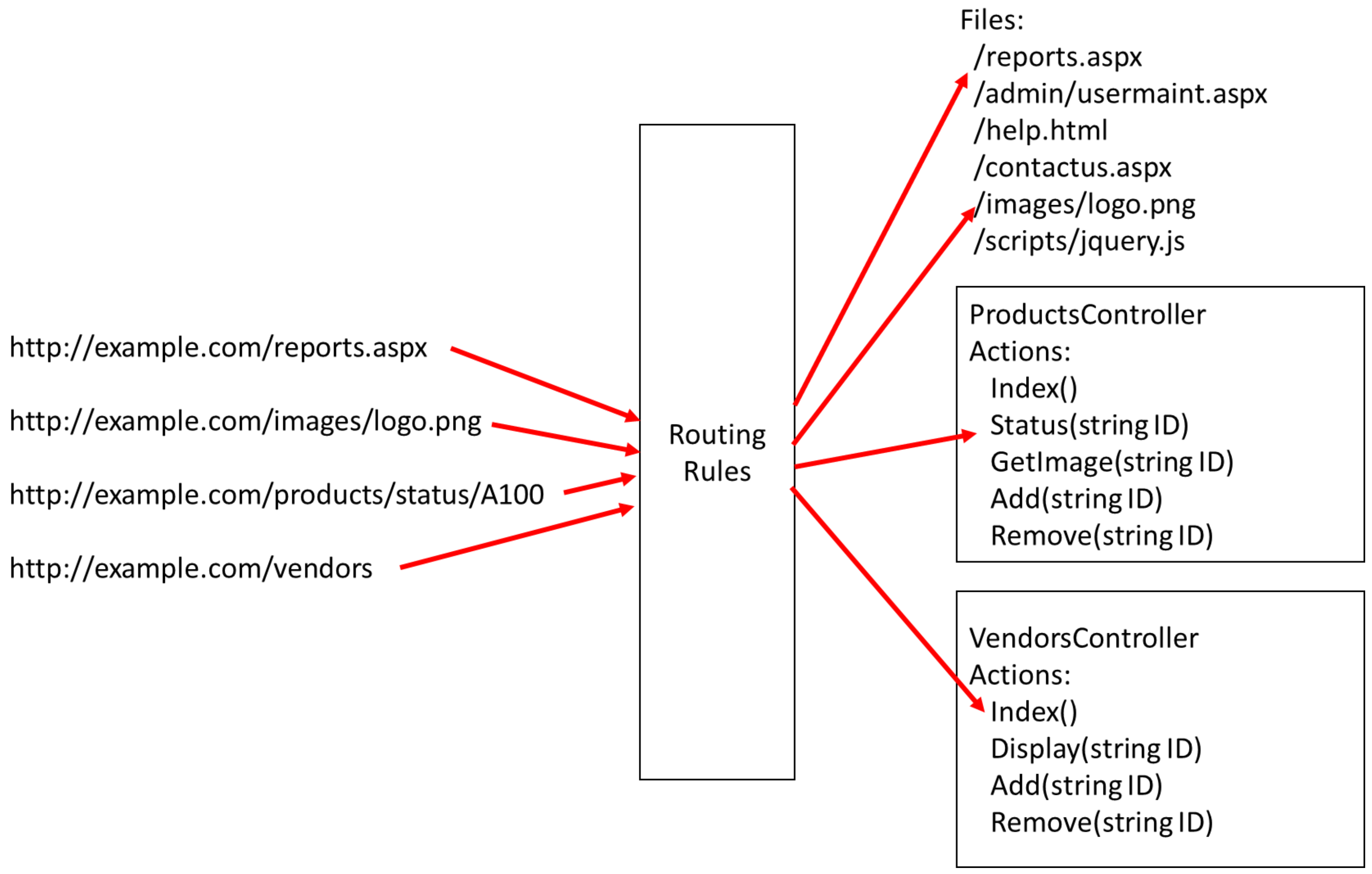
```
public class Startup
{
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IConfiguration Configuration { get; }

    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)...

    // This method gets called by the runtime. Use this method to configure the HTTP request pip
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)...
}
```

Routing


Files:
/reports.aspx
/admin/usermaint.aspx
/help.html
/contactus.aspx
/images/logo.png
/scripts/jquery.js
http://example.com/reports.aspx
http://example.com/images/logo.png
http://example.com/products/status/A100
http://example.com/vendors
Routing Rules
ProductsController
Actions:
Index()
Status(string ID)
GetImage(string ID)
Add(string ID)
Remove(string ID)
VendorsController
Actions:
Index()
Display(string ID)
Add(string ID)
Remove(string ID)

Routing

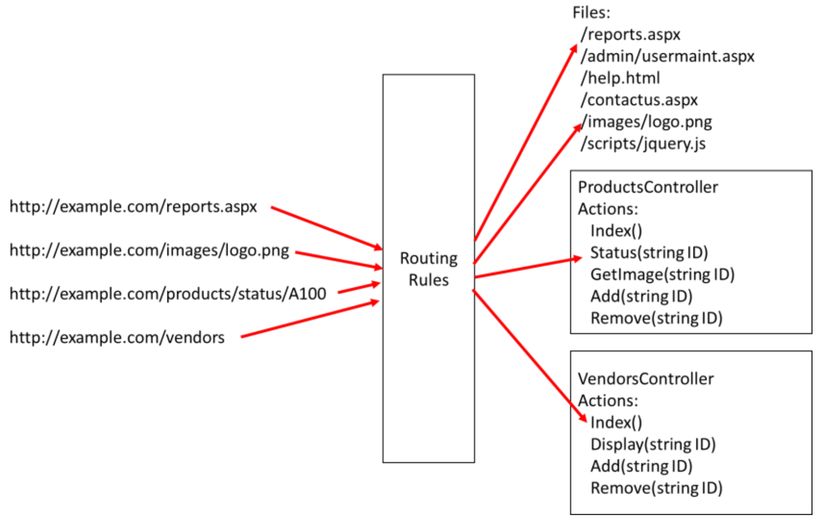
- .NET Full Framework
  - Via code in Global.asax or a class file linked from Global.asax (/App_Start/RouteConfig.cs)

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
 );
```

  - Via Controller or Action markup
- NET Core
  - Part of the ASP.NET Core startup process. (/StartUp.cs – Configure method)

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

  - Via Controller or Action markup



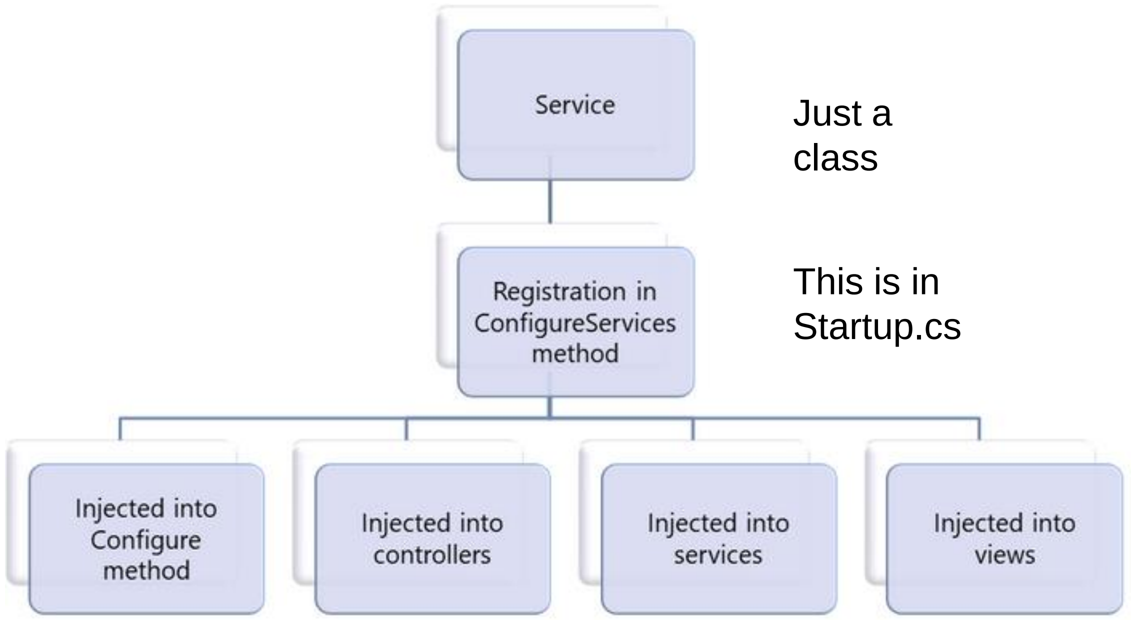
Dependency Injection

- .NET Full Framework
  - Write your own code, create interfaces, etc.
  - You have to figure where and how…
- .NET Core
  - Part of the ASP.NET Core startup process.
  - Built-in for:
    - Controllers, Views, etc.

Middleware and Services

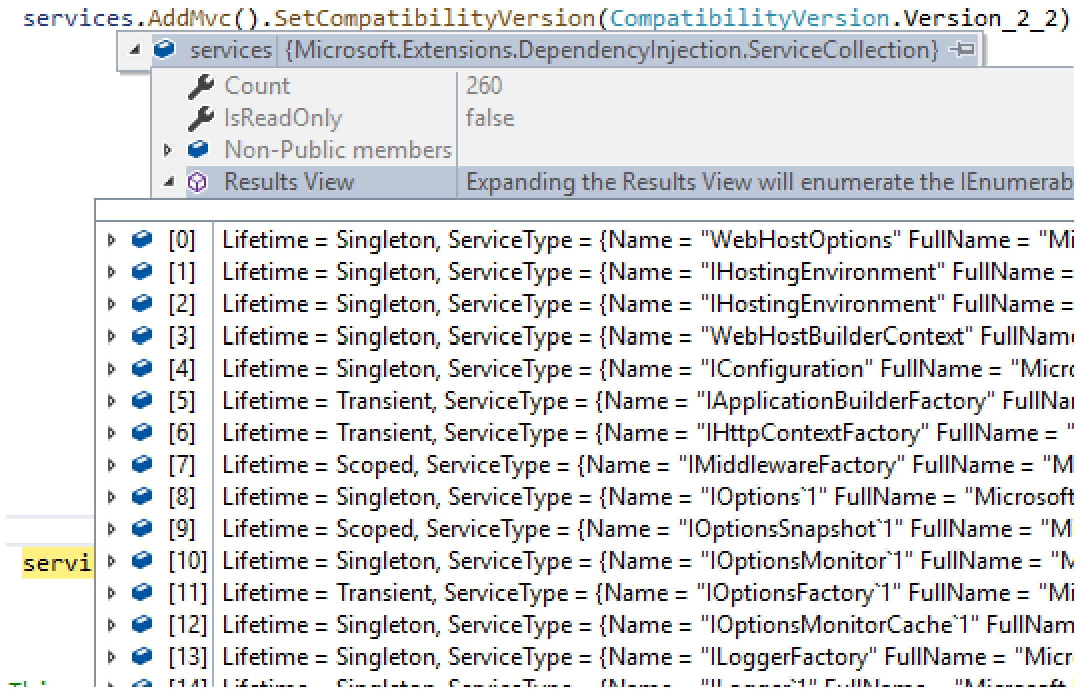
- .NET Full Framework
  - Write your own code, create interfaces, etc.
  - You have to figure where and how…
- .NET Core
  - Part of the ASP.NET Core startup process.
  - Built-in for:
    - Controllers, Views, … everything?
    - 260 Services just for a new MVC project.

Middleware and Services


Service
Registration in ConfigureServices method
Injected into Configure method
Injected into controllers
Injected into services
Injected into views

Just a class

This is in Startup.cs



```
services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2)
  services {Microsoft.Extensions.DependencyInjection.ServiceCollection}
    Count            260
    IsReadOnly       false
    Non-Public members
    Results View     Expanding the Results View will enumerate the IEnumerab
servi
  [0]  Lifetime = Singleton, ServiceType = {Name = "WebHostOptions" FullName = "Mi
  [1]  Lifetime = Singleton, ServiceType = {Name = "IHostingEnvironment" FullName =
  [2]  Lifetime = Singleton, ServiceType = {Name = "IHostingEnvironment" FullName =
  [3]  Lifetime = Singleton, ServiceType = {Name = "WebHostBuilderContext" FullNam
  [4]  Lifetime = Singleton, ServiceType = {Name = "IConfiguration" FullName = "Micr
  [5]  Lifetime = Transient, ServiceType = {Name = "IApplicationBuilderFactory" FullNa
  [6]  Lifetime = Transient, ServiceType = {Name = "IHttpContextFactory" FullName = "
  [7]  Lifetime = Scoped, ServiceType = {Name = "IMiddlewareFactory" FullName = "M
  [8]  Lifetime = Singleton, ServiceType = {Name = "IOptions`1" FullName = "Microsoft
  [9]  Lifetime = Scoped, ServiceType = {Name = "IOptionsSnapshot`1" FullName = "M
  [10] Lifetime = Singleton, ServiceType = {Name = "IOptionsMonitor`1" FullName = "M
  [11] Lifetime = Transient, ServiceType = {Name = "IOptionsFactory`1" FullName = "Mi
  [12] Lifetime = Singleton, ServiceType = {Name = "IOptionsMonitorCache`1" FullNam
  [13] Lifetime = Singleton, ServiceType = {Name = "ILoggerFactory" FullName = "Micr
```

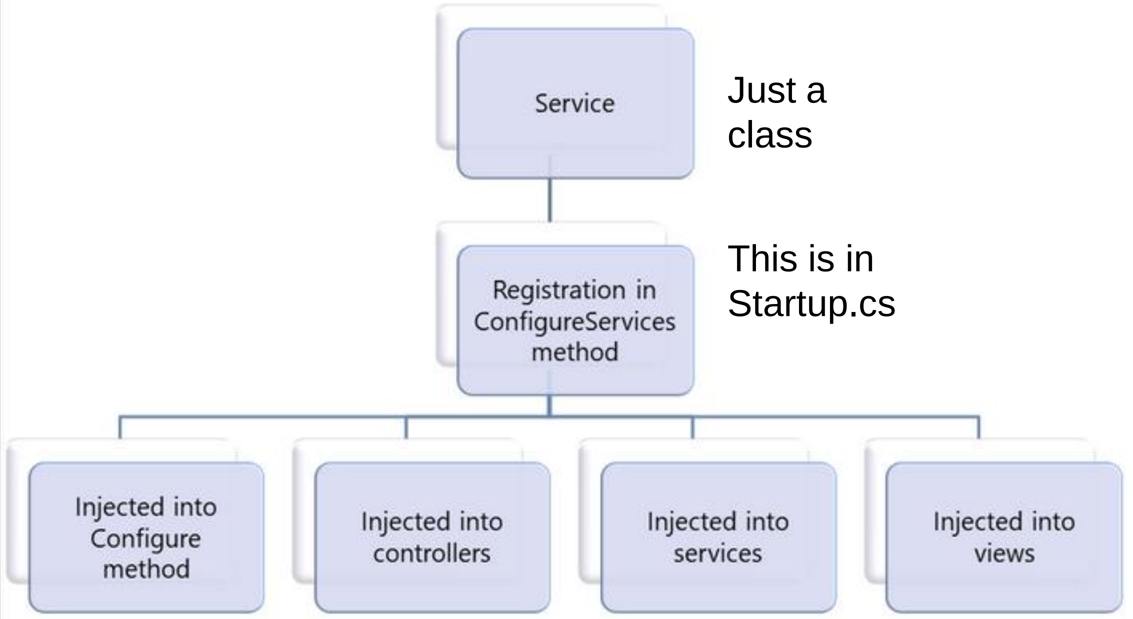
Middleware and Services

- Simple steps
  - Create the Service (a simple class)

    ```
    public interface IMyService { … }
    public class MyService : IMyService { … }
    ```
  - Register the service in Startup.cs

    ```
    services.AddSingleton<IMyService, MyService>();
    ```
  - Inject the service via a constructor

    -
      ```
      public class SomeClass
      {
          private IMyService _myService;
          public HomeController(IMyService myService)
          {
              _myService = myService;
          }
      ```
  - Inject the service into a View
    - @inject <type> <instanceName>
    - @instanceName.method()


Service
Just a class
Registration in ConfigureServices method
This is in Startup.cs
Injected into Configure method
Injected into controllers
Injected into services
Injected into views

- AddSingleton - Instantiates once in the application's lifetime
- AddScoped - Instantiates once per request made to the server
- AddTransient - Instantiates every single time the service is injected

Entity Framework

- Both are open source and in GitHub.
- Entity Framework (EF 1, 2, 3, 4, 5, 6)
  - Works only with Full Framework
  - EF6 continues to be a supported product, and will continue to see bug fixes and minor improvements.
  - Download from Nuget as needed.
  - Code first.
  - Model First and Design First with graphical designers/wizards.

- Entity Framework Core (1.0, 1.1, 2.0, 2.1, 2.2, 3.0 preview)
  - Works with Full Framework and Core.
  - Download from Nuget as needed. Included with ASP.NET Core templates.
  - Code first.
  - No graphical designers/wizards, so no Design First and only a PowerShell “database first”. No update model from database.
  - No Entity SQL, No “magic” auto created databases

https://github.com/aspnet/EntityFramework6 https://github.com/aspnet/EntityFrameworkCore
https://docs.microsoft.com/en-us/ef/efcore-and-ef6/

Entity Framework

- Entity Framework Core (1.0, 1.1, 2.0, 2.1, 2.2, 3.0 preview)
  - Code first. But without much of the “magic” and “wizards”.
  - Database First wizard is gone. There is a command line tool.

    ```
    Scaffold-DbContext
    "Server=(localdb)\mssqllocaldb;Database=Blogging;Trusted_Connection=True;"
    Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
    ```

    - You can also specify tables:

      ```
      -Tables Customers, Products
      ```

    - No update model from database.
  - No “magic” auto generated database.
  - Some of the model attributes are missing in Core, but can be duplicated using the Fluent API. (Index, Key with multiple columns)
  - Many features available as both PowerShell and “dotnet” console.
    - Add-Migration InitialCreate dotnet ef migrations add InitialCreate
    - *Part of the multiple platform way of doing things…*

Entity Framework

- New in Entity Framework Core
  - Constructors with parameters *(Docs)*
  - Property value conversions *(Docs)*
  - Query types (Mapped types with no keys) *(Docs)*
  - Shadow properties *(Docs)*
  - Alternate keys *(Docs)*
  - Global query filters *(Docs)*
  - Mixed client/server evaluation *(Docs)*
  - Loading related data: Eager loading for derived types *(Docs)*

Other MVC Goodies

- New in Core MVC
  - Controllers prebuilt for dependency injection
  - Built in support for that annoying “We use cookies stuff”!

```
options.CheckConsentNeeded = context => true;
 app.UseCookiePolicy()
```

Where to go next…

- We have a course for that!
  - – The “D” version of 20486 has been updated for ASP.NET Core MVC
    - If you have taken a previous version from MAX, you can take this for free! (The famous MAX Retake Policy!)

References

- Migrate from ASP.NET to ASP.NET Core
  https://docs.microsoft.com/en-us/aspnet/core/migration/proper-to-2x/?view=aspnetcore-2.2

Instruction from the Trenches

Mike Smith

- Email address: mike@maxtrain.com
- Twitter: @TechTrainNotes
- Blog: TechTrainingNotes.blogspot.com

max technical training