.NET Core, ASP.NET Core, and ASP.NET Core MVC

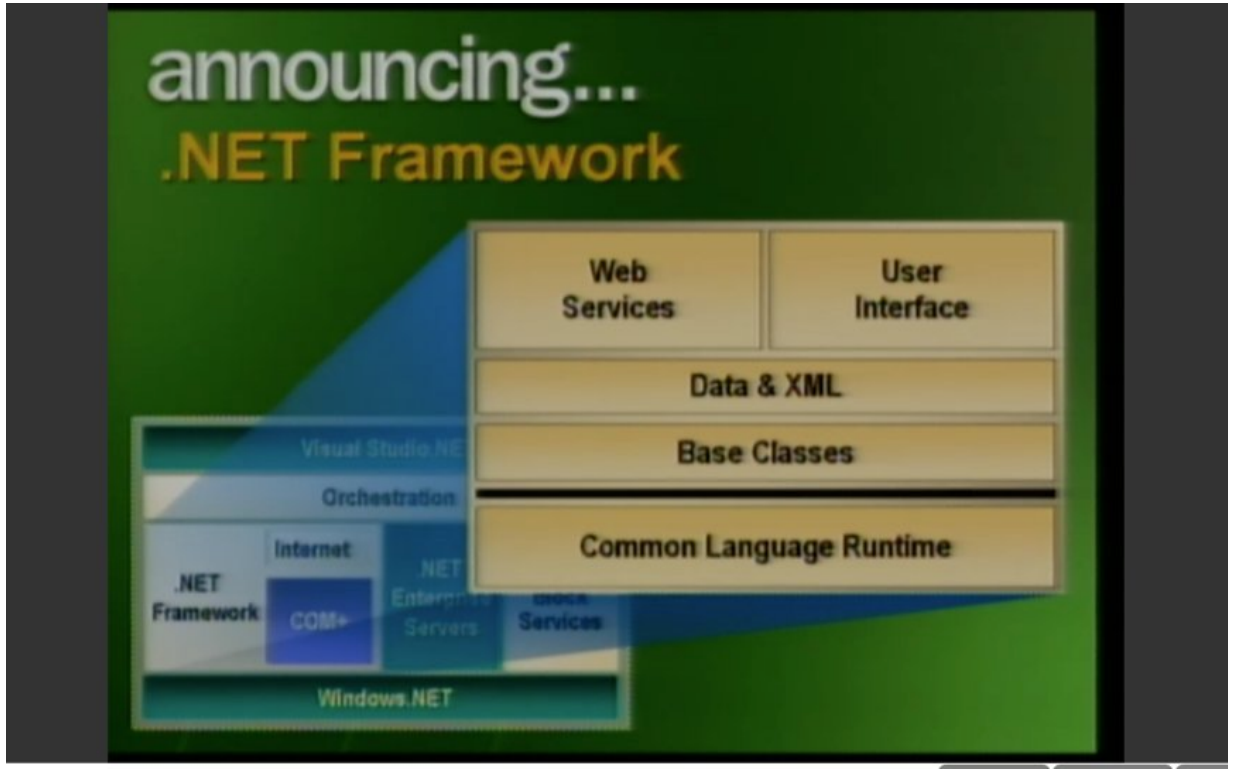
brave new world

Outline

- Motivation
- .NET Core
- ASP.NET Core
- ASP.NET Core MVC

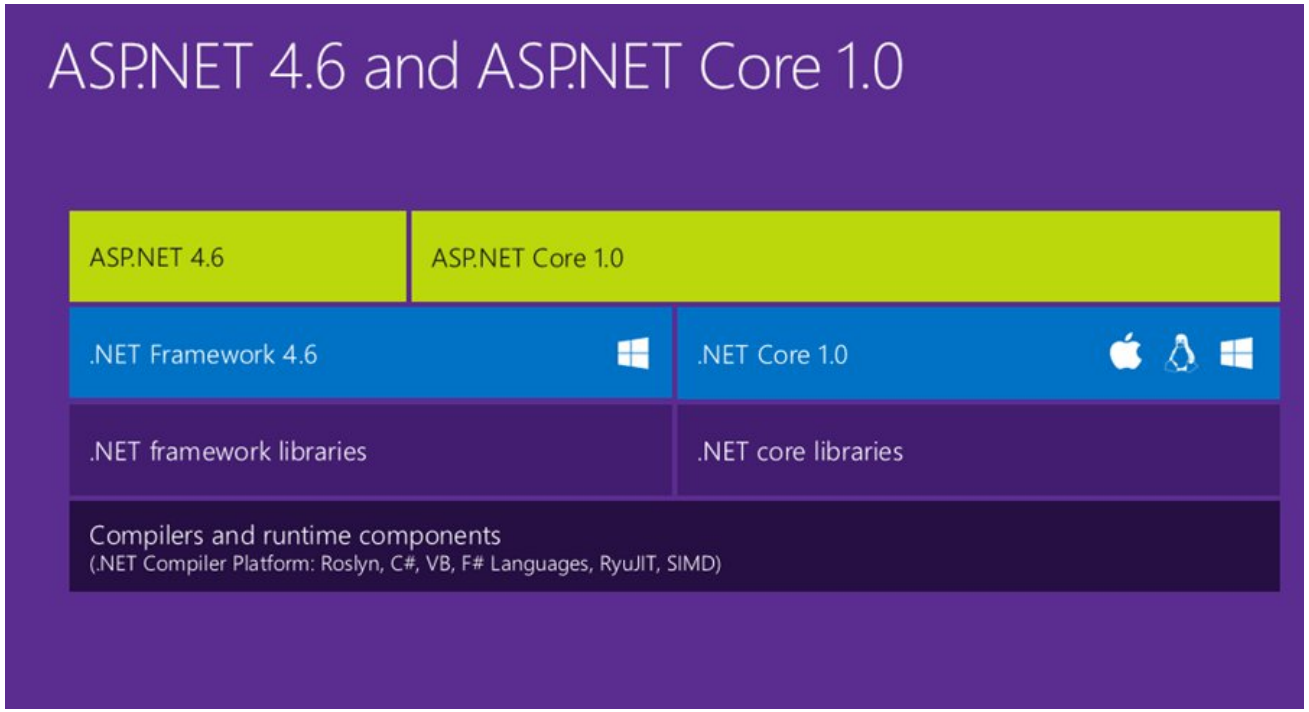
Motivation

- .NET has a strong history
  - Very popular
  - Lots of investments
- There is more than just Windows
  - Many more platforms, devices, and clouds
- .NET is evolving
  - .NET needs to learn to run in more places
  - .NET needs modern tooling



.NET runtimes target a platform

- .NET Framework
  - Windows-only
- .NET Core
  - Cross-platform runtime
- .NET Native (UWP)
- Mono
- Windows Phone
- More…



.NET Core : next gen .NET for server apps

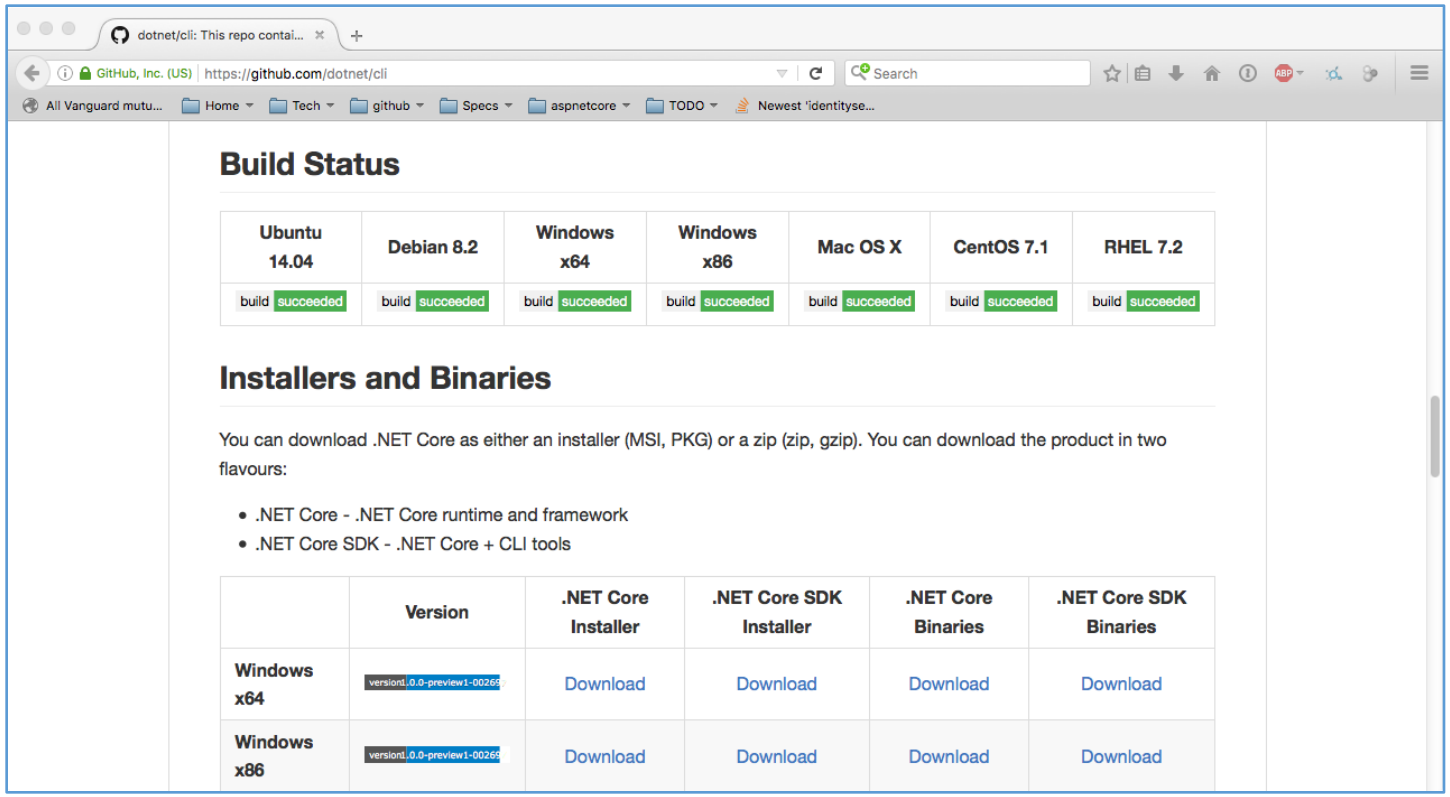
- Cross platform
  - Windows, Linux, Mac, FreeBSD
- Portable
  - Can be ~/bin deployed
  - Can be user or machine installed as well
- Open source
  - https://github.com/dotnet/coreclr
  - Contains core runtime and mscorlib (e.g. GC, JIT, BCL)
  - Does not contain many frameworks (e.g. WCF, WPF)

Development ecosystem

- SDK
  - Command-line tooling (*dotnet*)
- Project system
  - File-system based project system (*project.json*)
- Runtime, libraries, and packaging
  - NuGet-focused
- Editors/IDEs
  - Any text editor (VS Code, Emacs, Sublime, etc) and OmniSharp (OSS)
  - Visual Studio (Microsoft)
  - Project Rider (JetBrains)

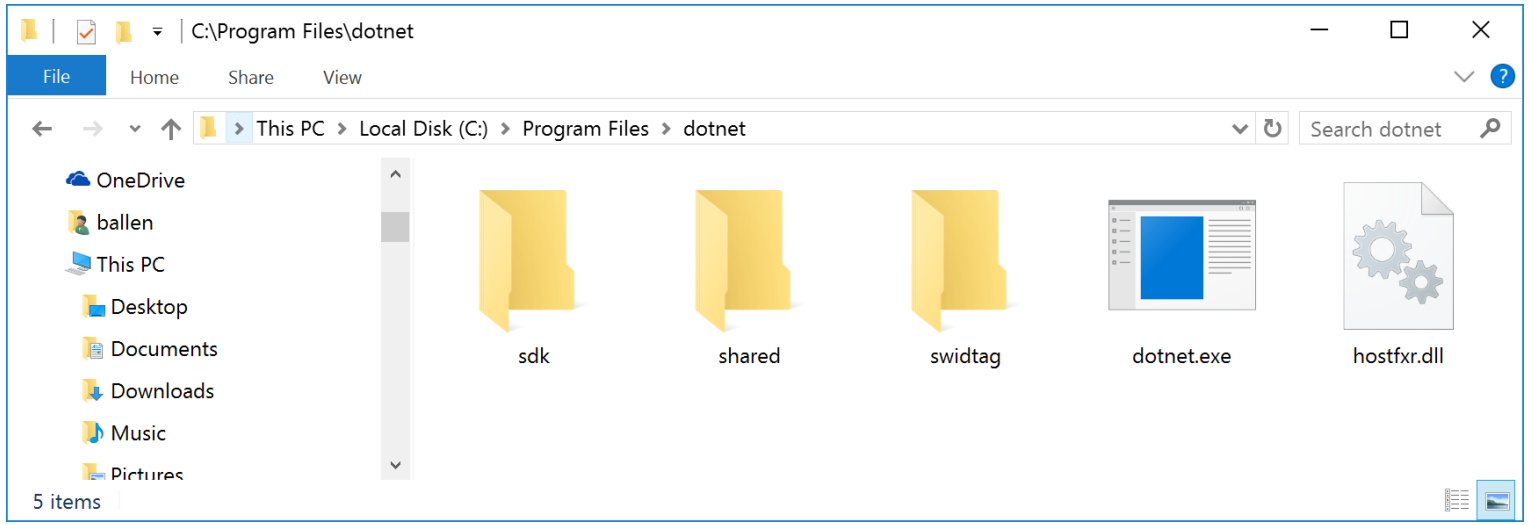
Installing .NET SDK

- Use nightly builds (until RC2 is released)
  - https://github.com/dotnet/cli



dotnet : command line tool

- Create new project
- Install NuGet dependencies
- Build application
- Load .NET and run application
- Package library
- Publish application



dotnet new

- Creates new project
  - program.cs
  - project.json
- Console-based application

C:\Windows\system32\cmd.exe

```
C:\demos\app>dotnet new
Created new C# project in C:\demos\app.

C:\demos\app>ls
Program.cs
project.json

C:\demos\app>
```

```
using System;

namespace ConsoleApplication
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

project.json

- Project type
  - Application or library
- Dependencies
  - Primarily from NuGet
- Target framework(s)
  - Target framework moniker (TFM)

```
{
  "version": "1.0.0-*",
  "buildOptions": {
    "emitEntryPoint": true
  },
  "dependencies": {
    "Microsoft.AspNetCore.Mvc": "1.0.0-rc2-*"
  },
  "frameworks": {
    "net46" : {},
    "netcoreapp1.0": {
      "dependencies": {
        "Microsoft.NETCore.App": {
          "type": "platform",
          "version": "1.0.0-rc2-3002659"
        }
      }
    }
  }
}
```

.NET platform standard

- Identifier (TFM) for required framework
  - Replacement for PCL platform versioning nightmare
- Libraries target an expected API from framework
  - "netstandard1.0", "netstandard1.1", …, "netstandard1.5"
  - Can use libraries from earlier .NET Standard version
- Applications target a specific platform (and thus framework)
  - "net451", "net452", "net46", "net461", "netcoreapp1.0", etc…
  - Platforms support a specific .NET Standard version

https://github.com/dotnet/corefx/blob/master/Documentation/architecture/net-platform-standard.md

Platform support for .NET Standard

Target Platform Name	Alias						
.NET Platform Standard	netstandard	1.0	1.1	1.2	1.3	1.4	1.5
.NET Core	netcoreapp	→	→	→	→	→	1.0
.NET Framework	net	→	→	→	→	→	4.6.2
		→	→	→	→	4.6.1	
		→	→	→	4.6		
		→	→	4.5.2			
		→	→	4.5.1			
		→	4.5				
Universal Windows Platform	uap	→	→	→	→	10.0	
Windows	win	→	→	8.1			
		→	8.0				
Windows Phone	wpa	→	→	8.1			
Windows Phone Silverlight	wp	8.1					
		8.0					
Mono/Xamarin Platforms		→	→	→	→	→	*
Mono		→	→	*			

dotnet restore

- Downloads NuGet dependencies
  - Might need a local nuget.config to target nightly builds from myget.org

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <packageSources>
    <!--To inherit the global NuGet package sources remove the <clear/> line below -->
    <clear />
    <add key="dotnet-core" value="https://dotnet.myget.org/F/dotnet-core/api/v3/index.json" />
    <add key="AspNetCI" value="https://www.myget.org/F/aspnetcirelease/api/v3/index.json" />
  </packageSources>
</configuration>
```

- Builds project.json.lock
  - Snapshot of dependency versions
  - Needed to load application

dotnet build / dotnet run / dotnet app.dll

- Builds project, or builds and runs application
  - -c indicates configuration (release/debug)
  - -f indicates framework to target
  - -v emits verbose log output
- Binaries output to ~/bin/*<configuration>*/*<framework>* folder

C:\Windows\system32\cmd.exe

```
c:\demos\app>dotnet -v run
Telemetry is: Enabled
Project app (.NETCoreApp,Version=v1.0) was previously compiled. Skipping compilation.
Running C:\Program Files\dotnet\dotnet.exe exec --additionalprobingpath C:\Users\ballen\.
nuget\packages c:\demos\app\bin\Debug\netcoreapp1.0\app.dll
Process ID: 4880
Hello World!

c:\demos\app>
```

ASP.NET Core

- The new pipeline
- Middleware
- Dependency Injection
- Configuration

Motivation

- Modern web stack
  - Modern package system (NuGet)
  - Lightweight/composable runtime
  - Dependency injection
  - Flexible configuration/deployment

applicationHost.config - Microsoft Visual Studio

```
240         -->
241         <globalModules>
242             <add name="UriCacheModule" image="%windir%\System32\inetsrv\cachuri.dll" />
243             <add name="FileCacheModule" image="%windir%\System32\inetsrv\cachfile.dll" />
244             <add name="TokenCacheModule" image="%
245             <add name="HttpCacheModule" image="%w
246             <add name="StaticCompressionModule" i
247             <add name="DefaultDocumentModule" ima
248             <add name="DirectoryListingModule" im
249             <add name="ProtocolSupportModule" ima
250             <add name="StaticFileModule" image="%
251             <add name="AnonymousAuthenticationMod
252             <add name="CertificateMappingAuthenti
253             <add name="BasicAuthenticationModule"
254             <add name="WindowsAuthenticationModul
255             <add name="IISCertificateMappingAuthe
256             <add name="RequestFilteringModule" im
257             <add name="CustomErrorModule" image="
258             <add name="HttpLoggingModule" image="
259             <add name="IsapiModule" image="%windi
260             <add name="IsapiFilterModule" image="
261             <add name="ManagedEngineV4.0_32bit" i
262             <add name="ApplicationInitializationM
263             <add name="WebSocketModule" image="%w
264             <add name="ManagedEngineV4.0_64bit" i
```

applicationHost.config - Microsoft Visual Studio

```
805         <handlers accessPolicy="Read, Script">
806             <add name="ISAPI-dll" path="*.dll" verb="*" modules="IsapiModule" resourceType="File"
807                 <add name="AXD-ISAPI-4.0_64bit" path="*.axd" verb="GET,HEAD,POST,DEBUG" modules="
808                 <add name="PageHandlerFactory-ISAPI-4.0_64bit" path="*.aspx" verb="GET,HEAD,POST,
809                 <add name="SimpleHandlerFactory-ISAPI-4.0_64bit" path="*.ashx" verb="GET,HEAD,POS
810                 <add name="WebServiceHandlerFactory-ISAPI-4.0_64bit" path="*.asmx" verb="GET,HEAD
811                 <add name="HttpRemotingHandlerFactory-rem-ISAPI-4.0_64bit" path="*.rem" verb="GET
812                 <add name="HttpRemotingHandlerFactory-soap-ISAPI-4.0_64bit" path="*.soap" verb="G
813                 <add name="aspq-ISAPI-4.0_64bit" path="*.aspq" verb="*" modules="IsapiModule" scr
814                 <add name="cshtm-ISAPI-4.0_64bit" path="*.cshtm" verb="GET,HEAD,POST,DEBUG" modul
815                 <add name="cshtml-ISAPI-4.0_64bit" path="*.cshtml" verb="GET,HEAD,POST,DEBUG" moc
816                 <add name="vbhtm-ISAPI-4.0_64bit" path="*.vbhtm" verb="GET,HEAD,POST,DEBUG" modul
817                 <add name="vbhtml-ISAPI-4.0_64bit" path="*.vbhtml" verb="GET,HEAD,POST,DEBUG" moc
818                 <add name="TraceHandler-Integrated-4.0" path="trace.axd" verb="GET,HEAD,POST,DEBU
819                 <add name="WebAdminHandler-Integrated-4.0" path="WebAdmin.axd" verb="GET,DEBUG" t
820                 <add name="AssemblyResourceLoader-Integrated-4.0" path="WebResource.axd" verb="GE
821                 <add name="PageHandlerFactory-Integrated-4.0" path="*.aspx" verb="GET,HEAD,POST,D
822                 <add name="SimpleHandlerFactory-Integrated-4.0" path="*.ashx" verb="GET,HEAD,POST
823                 <add name="WebServiceHandlerFactory-Integrated-4.0" path="*.asmx" verb="GET,HEAD,
824                 <add name="HttpRemotingHandlerFactory-rem-Integrated-4.0" path="*.rem" verb="GET,
825                 <add name="HttpRemotingHandlerFactory-soap-Integrated-4.0" path="*.soap" verb="GE
826                 <add name="aspq-Integrated-4.0" path="*.aspq" verb="GET,HEAD,POST,DEBUG" type="Sy
827                 <add name="cshtm-Integrated-4.0" path="*.cshtm" verb="GET,HEAD,POST,DEBUG" type="
828                 <add name="cshtml-Integrated-4.0" path="*.cshtml" verb="GET,HEAD,POST,DEBUG" type
829                 <add name="vbhtm-Integrated-4.0" path="*.vbhtm" verb="GET,HEAD,POST,DEBUG" type="
```

Ready Ln 805 Col 18 Ch 18 INS

"Empty Web Application"

Web.config*

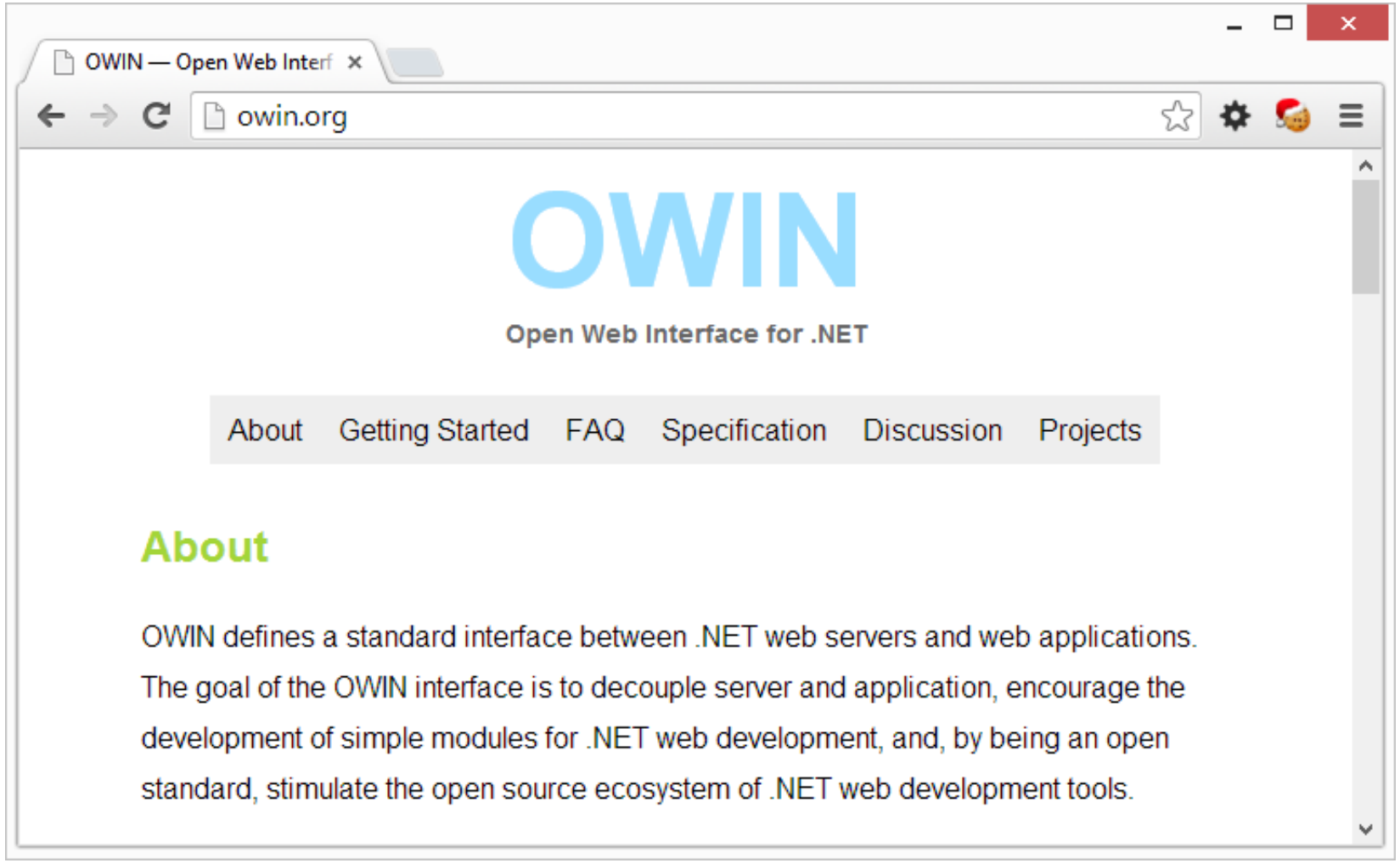
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <configuration>
3    <configSections>
4      <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->
5      <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture
6    </configSections>
7    <connectionStrings>
8      <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MvcApplication1-2013122112
9        providerName="System.Data.SqlClient" />
10   </connectionStrings>
11   <appSettings>
12     <add key="webpages:Version" value="3.0.0.0" />
13     <add key="webpages:Enabled" value="false" />
14     <add ke
15     <add ke
16   </appSetti
17   <system.we
18     <authent
19     <compila
20     <httpRu
21   </system.
22   <system.we
23     <module
24       <remo
25     </module
26   </system.
27   <runtime>
28     <assembl
29       <depe
30         <as
31         <bi
32       </dep
33       <depe
34         <as
35         <bi
36       </dep
37       <depe
38         <as
39         <bi
40       </dep
41       <depe
42         <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
43         <bindingRedirect oldVersion="1.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
44       </dependentAssembly>
45     </assemblyBinding>
46   </runtime>
47   <entityFramework>
48     <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
49       <parameters>
50         <parameter value="v11.0" />
51       </parameters>
52     </defaultConnectionFactory>
53     <providers>
54       <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
55     </providers>
```

Solution 'MvcApplication' (1 project)

- MvcApplication
  - Properties
  - References
    - Microsoft.CSharp
    - Microsoft.Web.Infrastructure
    - Microsoft.Web.Mvc.FixedDisplayModes
    - Newtonsoft.Json
    - System
    - System.ComponentModel.DataAnnotations
    - System.Configuration
    - System.Web.Http.WebHost
    - System.Web.Mvc
    - System.Web.Razor
    - System.Web.Routing
    - System.Web.Services
    - System.Web.WebPages
    - System.Web.WebPages.Deployment
    - System.Web.WebPages.Razor
    - System.Xml
    - System.Xml.Linq

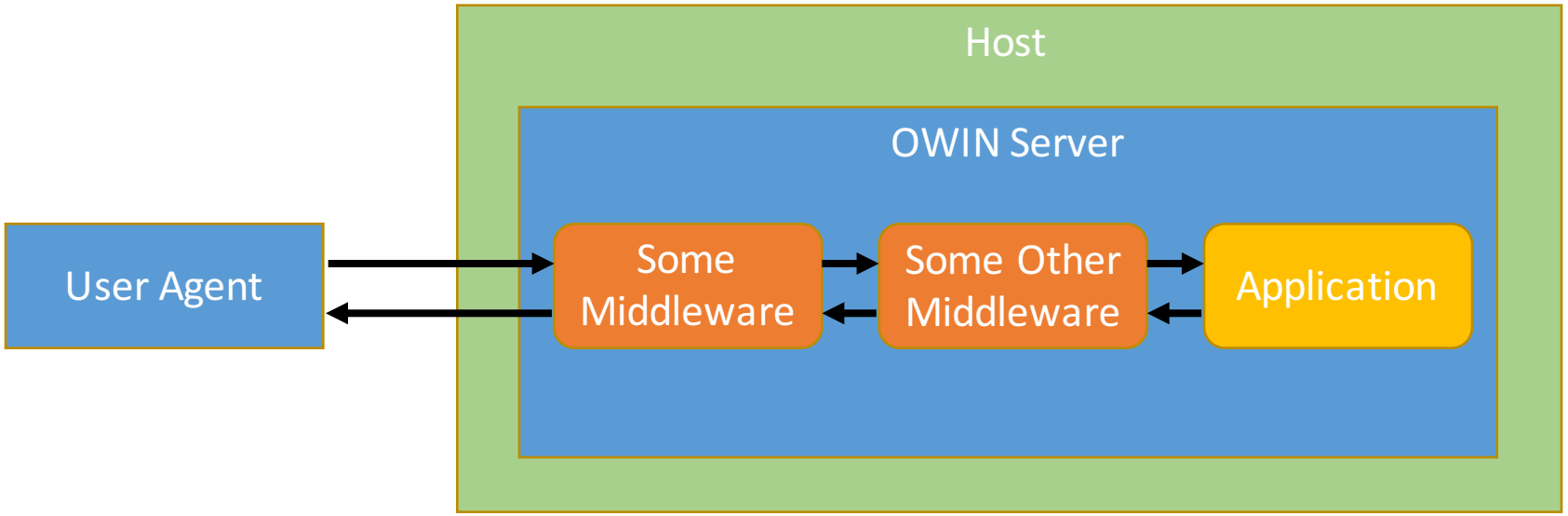
An example of a web server written with Node which responds with 'Hello World':

```
var http = require('http');

http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
}).listen(8124);

console.log('Server running at http://127.0.0.1:8124/');
```


OWIN — Open Web Interf
owin.org
OWIN
Open Web Interface for .NET
About
Getting Started
FAQ
Specification
Discussion
Projects
About
OWIN defines a standard interface between .NET web servers and web applications. The goal of the OWIN interface is to decouple server and application, encourage the development of simple modules for .NET web development, and, by being an open standard, stimulate the open source ecosystem of .NET web development tools.

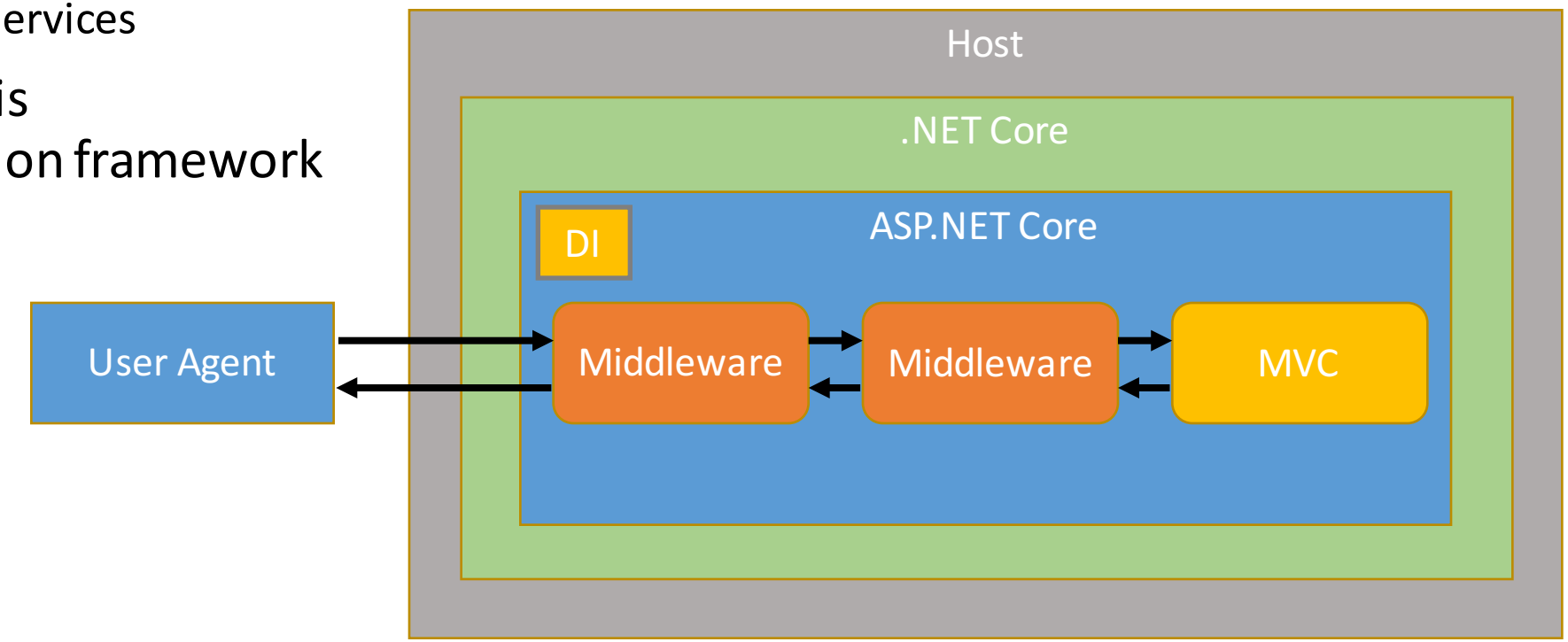
OWIN Middleware Architecture

- Middleware are linked components that process requests
- Application code targeting a framework

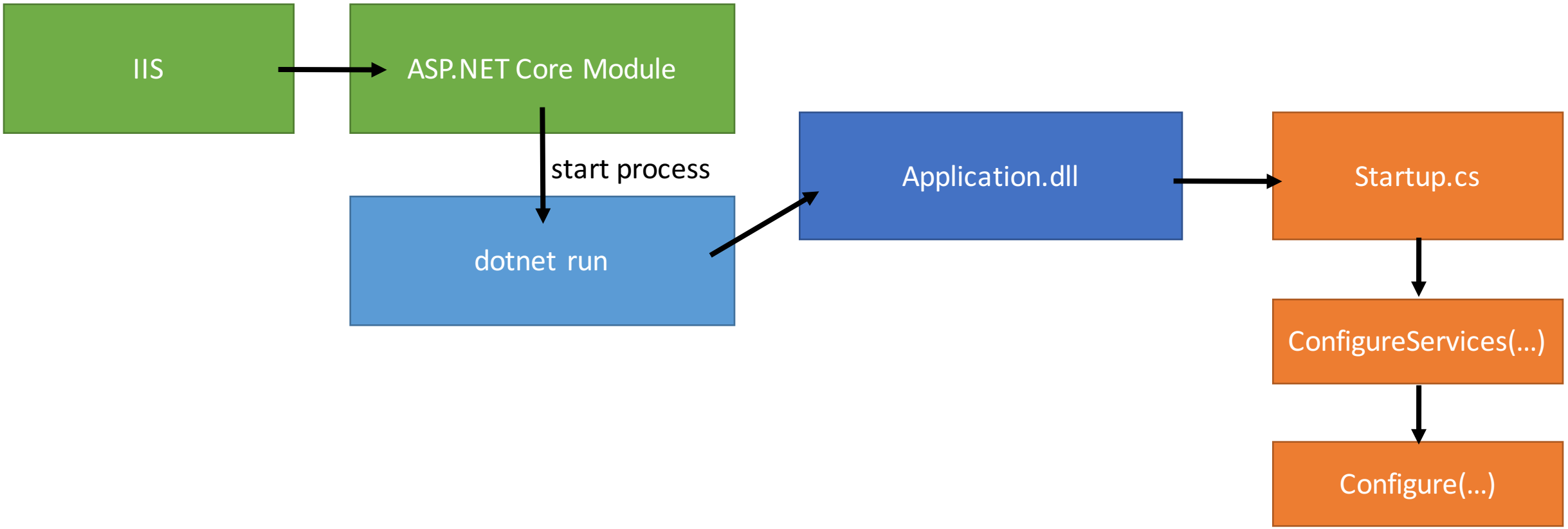


ASP.NET Core

- ASP.NET Core is HTTP pipeline implementation
  - sits on top of .NET Core
  - uses the middleware concept (but at a higher abstraction level than OWIN)
  - comes with its own server (Kestrel)
  - adds DI to provide services
- ASP.NET Core MVC is Microsoft's application framework



How ASP.NET Core Applications start

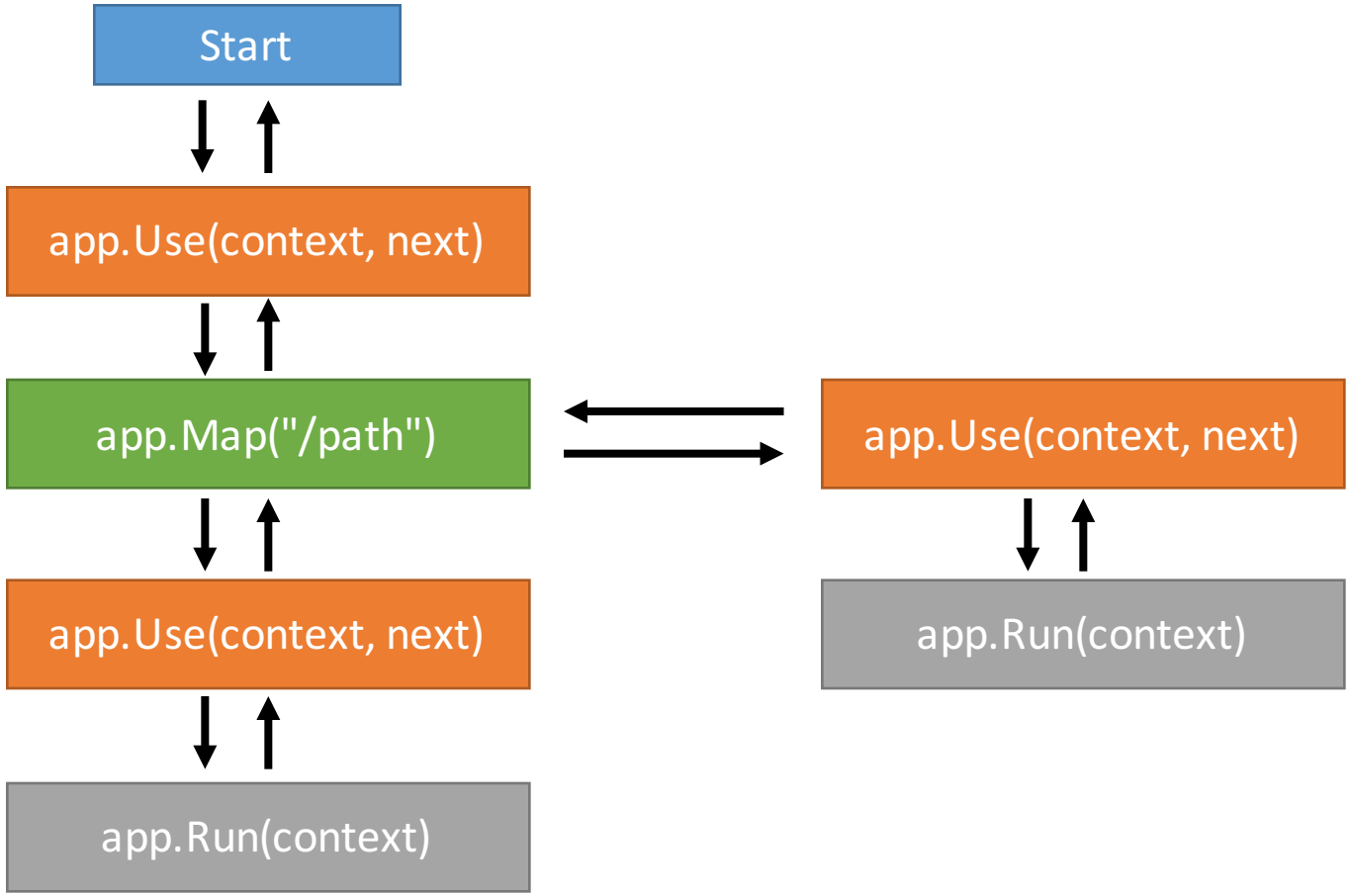


Loading ASP.NET Core

```
public class Program
{
    public static void Main()
    {
        var host = new WebHostBuilder()
            .UseKestrel()
            .UseIISIntegration()
            .UseStartup<Startup>()
            .Build();

        host.Run();
    }
}
```

```
public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        ...
    }
}
```

Pipeline primitives



Run

```
namespace Microsoft.AspNetCore.Builder
{
    public delegate Task RequestDelegate(HttpContext context);
}
```

```
app.Run(async context =>
{
    await context.Response.WriteAsync("Hello ASP.NET Core");
});
```

Map

```
app.Map("/hello", helloApp =>
{
    helloApp.Run(async (HttpContext context) =>
    {
        await context.Response.WriteAsync("Hello ASP.NET Core");
    });
});
```

Use

```
app.Use(async (context, next) =>
{
    if (!context.Request.Path.Value.EndsWith("/favicon.ico"))
    {
        Console.WriteLine("pre");
        Console.WriteLine(context.Request.Path);

        await next();

        Console.WriteLine("post");
        Console.WriteLine(context.Response.StatusCode);
    }
    else
    {
        await next();
    }
});
```

Middleware classes

```
app.UseMiddleware<InspectionMiddleware>();
```

```
public class InspectionMiddleware
{
    private readonly RequestDelegate _next;

    public InspectionMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public async Task Invoke(HttpContext context)
    {
        Console.WriteLine($"request: {context.Request.Path}");
        await _next(context);
    }
}
```

Dependency Injection

- Various places
  - Configure
  - Middleware classes
  - Higher-level frameworks (e.g. MVC controller)

- Host provided dependencies (e.g. *IHostingEnvironment, ILoggerFactory*)
- Dependencies set up in *ConfigureServices*

DI Examples

```
public class Startup
{
    public Startup(IHostingEnvironment environment)
    { /* stuff */ }

    public void ConfigureServices(IServiceCollection services)
    { /* register more stuff */ }

    public void Configure(IApplicationBuilder app, ILoggerFactory loggerFactory)
    {
       /* add middleware */
    }
}
```

Registering dependencies

- New instance "per call"

```
services.AddTransient<IMyCustomService, MyCustomService>();
```

- New instance per HTTP request

```
services.AddScoped<IMyCustomService, MyCustomService>();
```

- Singleton

```
services.AddSingleton<IMyCustomService, MyCustomService>();
```

Configuration

- web.config is no more

- New configuration system based on key/value pairs
  - command line
  - environment variables
  - JSON files
  - INI files
- Configuration can come from multiple sources
  - last source wins

Example

```
public class Startup
{
    public IConfiguration Configuration { get; set; }

    public Startup(IHostingEnvironment env)
    {
        Configuration = new ConfigurationBuilder()
                    .SetBasePath(env.ContentRootPath)
                    .AddJsonFile("config.json")
                    .AddJsonFile($"config.{env.EnvironmentName}.json", optional: true)
                    .AddEnvironmentVariables()
                    .Build();
    }

    // more
}
```

Using configuration

```
public class Startup
{
    IConfiguration _configuration;

    public Startup()
    {
        _configuration = new ConfigurationBuilder()
          ...
          .Build();
    }

    public void Configure(IApplicationBuilder app)
    {
        var copyright = new Copyright
        {
            Company = _configuration.Get("copyright_company"),
            Year = _configuration.Get("copyright_year")
        };

        app.Run(async (context) =>
        {
            await context.Response.WriteAsync($"Copyright {copyright.Year}, {copyright.Company}");
        });
    }
}
```

```
{
  "copyright": {
    "year": "2015",
    "company": "Foo Industries"
  }
}
```

ASP.NET Core MVC

- Packaging
- Middleware
- Routing and action selection
- Controller initialization
- Model binding changes
- Razor
- Filters
- APIs
- Error handling

Packaging

- MVC is packaged entirely as a NuGet
  - Microsoft.AspNetCore.Mvc

```
{
  "dependencies": {
    "Microsoft.AspNetCore.Mvc": "1.0.0-rc2-*",
    "Microsoft.AspNetCore.Server.Kestrel": "1.0.0-rc2-*"
  }
}
```

Middleware

- MVC is configured as middleware
  - In ConfigureServices via AddMvc
  - In Configure via UseMvc

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
    }

    public void Configure(IApplicationBuilder app)
    {
        app.UseMvc();
    }
}
```

Overriding default settings

- Delegate callback param used to override defaults
  - Also fluent API on result from AddMvc()

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc(mvc =>
        {
            mvc.Filters.Add(...);
            mvc.ViewEngines.Add(...);
            mvc.InputFormatters.Add(...);
            mvc.OutputFormatters.Add(...);
        });
    }
}
```

Routing

- Routes configured via UseMvc
  - RouteParameters.Optional from MVC 5 removed

```
public void Configure(IApplicationBuilder app)
{
    app.UseMvc(routes =>
    {
        routes.MapRoute("old_default",
            "{controller}/{action}",
            new {
                controller = "Home", action="Index"
            });

        routes.MapRoute("new_default",
            "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Controllers

- Controller base class still provided
  - Action results now implement IActionResult
  - Controller base provides many helpers to create action results
    - View(), PartialView(), Content(), Json(), Ok(), Created(), HttpNotFound(), HttpUnauthorized(), HttpBadRequest(), File(), PhysicalFile(), Redirect(), RedirectPermanent()

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

Attribute routing

- Attribute routing enabled by default

```
public class HomeController : Controller
{
    // ~/ or ~/hello-world
    [Route("/")]
    [Route("/hello-world")]
    public IActionResult Index()
    {
        return View();
    }
}
```

Attribute routing

- Attribute routing can be applied to class
- [controller] and [action] act as tokens

```
[Route("[controller]/[action]")]
public class HomeController : Controller
{
    // ~/Home/Index
    public IActionResult Index()
    {
        return View();
    }
}
```

Combining Route attributes

- Route attributes inherit path
  - RoutePrefix from MVC 5 removed
- Can replace inherited path
  - If template starts with "/" or "~/"

```
[Route("[controller]")]
public class HomeController : Controller
{
    // ~/Home/hello
    [Route("hello")]
    public IActionResult Index()
    {
        return View();
    }

    // ~/hello
    [Route("/hello")]
    public IActionResult Index2()
    {
        return View();
    }
}
```

Route parameters

- [Route] allows parameters
  - With {param} syntax
- Supports filters
  - With {param:filter} syntax

```
[Route("[controller]/[action]")]
public class HomeController : Controller
{
    // GET ~/Home/Index
    public IActionResult Index()
    {
        return View();
    }

    // GET ~/Home/Index/5
    [Route("{id:int}")]
    public IActionResult Index(int id)
    {
        return View();
    }
}
```

HTTP method based routes

- HttpGet, HttpPost, HttpPut, HttpDelete, HttpPatch
  - Filter action method on request method
  - Build on [Route] semantics

```
[Route("[controller]/[action]")]
public class HomeController : Controller
{
    // GET ~/Home/Index
    [HttpGet]
    public IActionResult Index()
    {
        return View();
    }

    // ~/Submit
    [HttpPost("/Submit")]
    public IActionResult Submit()
    {
        return View();
    }
}
```

Areas

- Areas defined with the [Area] attribute
  - Used to match an {area} route param
  - Attribute routing allows [area] route token
- Views must still reside under ~/Areas/<area>/Views/<controller>

```
public void Configure(IApplicationBuilder app)
{
    app.UseMvc(routes =>
    {
        routes.MapRoute("new_default",
            "{area}/{controller=Home}/{action=Index}/{id?}");
    });
}
```

```
[Area("account")]
public class HomeController : Controller
{
    // ...
}
```

POCO controllers

- Controller classes can be POCO
  - Discovered in projects that reference Microsoft.AspNetCore.Mvc.*
  - Identified by "Controller" class name suffix
  - [NonController] disables

Dependency injection

- Can inject dependencies into controller ctor
- Special per-request types can be property injected with decorator attribute
  - ActionContext
  - ControllerContext

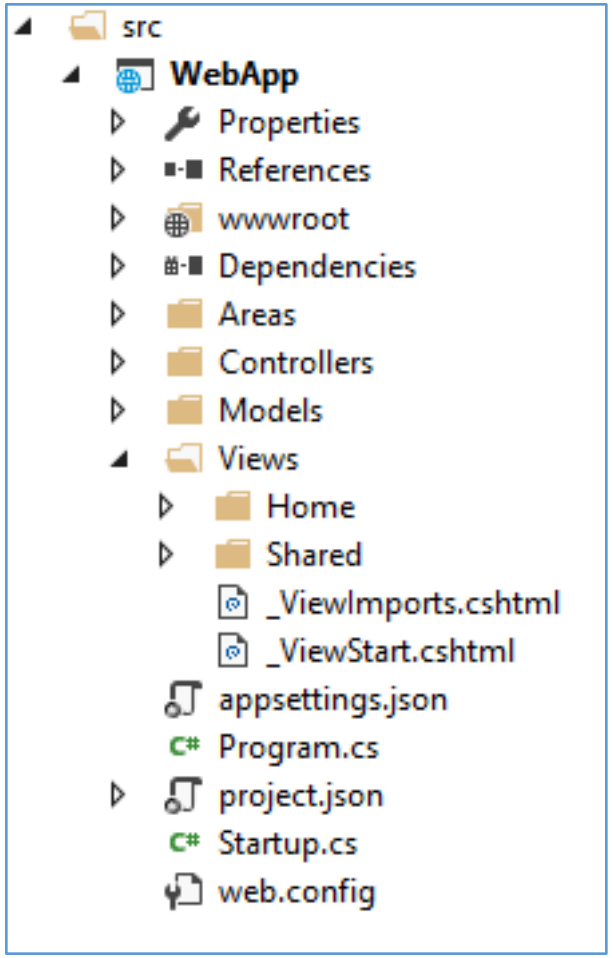
```
public class HomeController
{
    IHttpContextAccessor _accessor;

    public HomeController(IHttpContextAccessor accessor)
    {
        _accessor = accessor;
    }

    [ControllerContext]
    public ControllerContext ControllerContext { get; set; }

    // ...
}
```

Razor

- Shared config
  - _ViewStart and _ViewImports
- Chunks
  - @ directives
- TagHelpers
  - Like WebForms custom controls
- ViewComponents
  - Child action replacements

Shared razor configuration

- _ViewStart.cshtml still exists
  - Can now easily be put in application root
  - Layout assignment no longer is full path
- _ViewImports.cshtml is new
  - Allows for sharing @using, @addTagHelper chunks across views
  - Can be located in the same places as _ViewStart.cshtml



Razor directives (aka chunks)

- @model, @using, @section, @functions still exist
- @helper is gone
- @inject, @addTagHelper are new

- Also, @await Html.PartialAsync() is new

@inject

- Allows dependency injection into view
  - @inject *<type>* *<property>*

```
@using Microsoft.Framework.OptionsModel
@inject IOptions<MyConfig> Config

<h2>@Config.Options.SiteName</h2>
```

Tag helpers

- Like custom controls for MVC
  - Allow server-side code to inspect the element
  - Can modify attributes, tag, and/or contents
- @addTagHelper Namespace.ClassName, Assembly
  - Or @addTagHelper *, Assembly

```
@addTagHelper SpanTagHelper, YourProjectName

<span emoji="smile" />
```

Tag helper implementation

- TagHelper base class
  - Class name used to match element
- Override Process or ProcessAsync
  - Inspect element via TagHelperContext
  - Alter output via TagHelperOutput

```
public class SpanTagHelper : TagHelper
{
    public override void Process(
        TagHelperContext context, TagHelperOutput output)
    {
        if (context.AllAttributes.ContainsKey("emoji") &&
            "smile" == context.AllAttributes["emoji"].ToString())
        {
            output.Attributes.Add("title", "smile");
            output.Content.SetContent(" :) ");
            output.SelfClosing = false;
        }
    }
}
```

Tag helper implementation

- [TargetElement] can be used to match element
  - Attributes can be used to filter
- [HtmlAttributeName] will read incoming attribute
  - Will remove from output

```
[TargetElement("span", Attributes = "emoji")]
public class EmojiTagHelper : TagHelper
{
    [HtmlAttributeName("emoji")]
    public string Emoji { get; set; }

    public override void Process(
        TagHelperContext context, TagHelperOutput output)
    {
        if ("smile" == Emoji)
        {
            output.Attributes.Add("title", "smile");
            output.Content.SetContent(" :) ");
            output.SelfClosing = false;
        }
    }
}
```

MVC tag helpers

```
<a asp-controller="Manage" asp-action="Index">Manage Your Account</a>

<form asp-controller="Account" asp-action="LogOff" method="post"></form>

<environment names="Staging,Production">
   <h1>You're in production!</h1>
</environment>

<link rel="stylesheet"
      href="//ajax.aspnetcdn.com/ajax/bootstrap/3.0.0/css/bootstrap.min.css"
      asp-fallback-href="~/lib/bootstrap/css/bootstrap.min.css"
      asp-fallback-test-class="hidden"
      asp-fallback-test-property="visibility"
      asp-fallback-test-value="hidden" />

<script src="//ajax.aspnetcdn.com/ajax/jquery.validation/1.11.1/jquery.validate.min.js"
        asp-fallback-src="~/lib/jquery-validation/jquery.validate.js"
        asp-fallback-test="window.jquery && window.jquery.validator">
</script>
```

Validation tag helpers

```
<form asp-controller="Account" asp-action="ForgotPassword" method="post>
    <h4>Enter your email.</h4>

    <div asp-validation-summary="ValidationSummary.All" class="text-danger"></div>

    <div>
        <label asp-for="Email"></label>
        <input asp-for="Email" class="form-control" />
        <span asp-validation-for="Email" class="text-danger"></span>
    </div>
</form>
```

View components

- Replacement for child actions
  - Partial views still exist
- Allow for a partial view that runs controller-like code
  - Supports dependency injection

```
@Component.Invoke("Menu", 3)

Or

@await Component.InvokeAsync("Menu", 3)
```

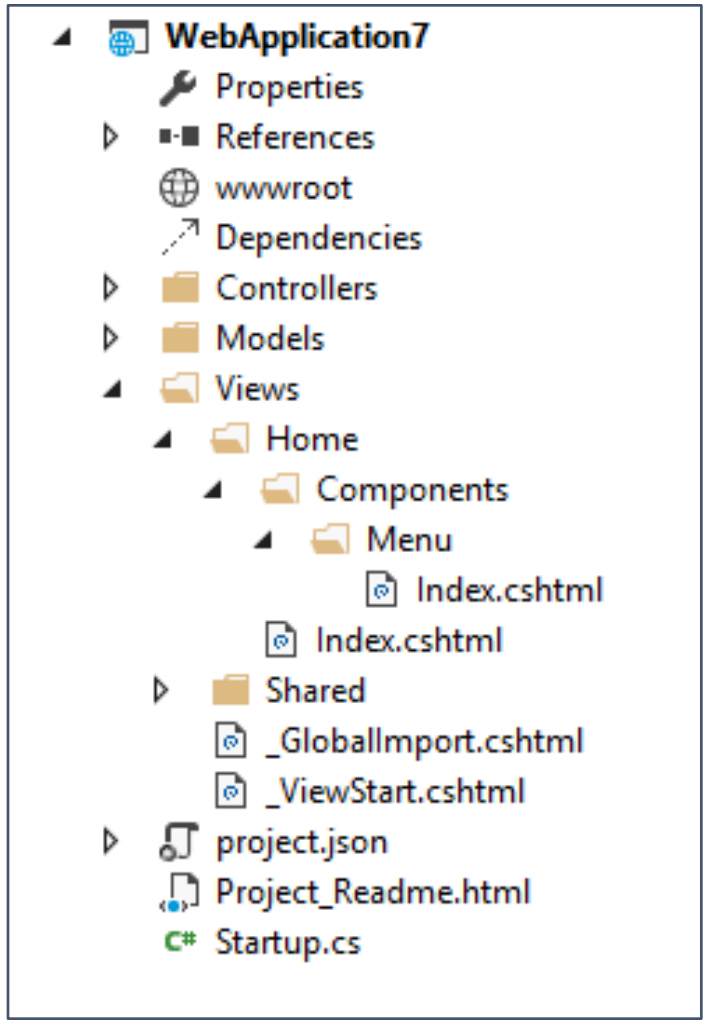
View components

- ViewComponent base class
  - Matched by class prefix
  - Or can use [ViewComponent] on POCO
- Implement Invoke or InvokeAsync
  - Returns IViewComponentResult or Task<IViewComponentResult>

```
public class MenuViewComponent : ViewComponent
{
    ICustomMenuService _menu;

    public MenuViewComponent(ICustomMenuService menu)
    {
        _menu = menu;
    }

    public IViewComponentResult Invoke(int depth)
    {
        var menuModel = _menu.GetMenu(depth);
        return View("Index", menuModel);
    }
}
```

View components

- View component views are under:
  - ~/Views/<controller>/Components/<component>
  - ~/Views/Shared/Components/<component>



Filters

- Dependency injection
- Resource filter
- Async support

TypeFilter

- Allows for filters that require dependency injection
  - Implemented via IFilterFactory

```
public class MyActionFilter : Attribute, IActionFilter
{
    private IHostingEnvironment _env;

    public MyActionFilter(IHostingEnvironment env)
    {
        _env = env;
    }

    // ...
}
```

```
[TypeFilter(typeof(MyFilter))]
public IActionResult Index()
{
    // ...
}
```

IResourceFilter

- Surrounds model binding, action, and result (including those filters)

```
public interface IResourceFilter : IFilter
{
   void OnResourceExecuting(ResourceExecutingContext context);
   void OnResourceExecuted(ResourceExecutedContext context);
}
```

- ResourceExecutingContext
  - Value providers, model binders, input formatters, validation providers
  - Can alter these on each request

Async filters

- All filters now have IAsync<*Filter*> support
  - Authorization, Resource, Action, Result, Exception
  - Pattern similar to middleware pipeline

```
public class MyResourceFilter : Attribute, IAsyncResourceFilter
{
    public async Task OnResourceExecutionAsync(
        ResourceExecutingContext context, ResourceExecutionDelegate next)
    {
        // pre
        var resourceExecutedContext = await next();
        // post
    }
}
```

Web API

- Formatters
  - Content negotiation
  - Format filters
  - XML support

Formatters

- Formatters have been split into two groups
  - Input formatters triggered via [FromBody]
  - Output formatters triggered via ObjectResult

Input formatters

- InputFormatter base class provides starting point
  - SupportedMediaTypes property used to match Content-Type header
  - [Consumes] filter can be used to limit formatters

Formatter	Content type	Comment
StringInputFormatter	text/plain	
JsonInputFormatter	application/json, text/json	
XmlSerializerInputFormatter	application/xml, text/xml	Not registered by default
XmlDataContractSerializerInputFormatter	application/xml, text/xml	Not registered by default

Output formatters

- ObjectResult chooses formatter from Accept header
  - OutputFormatter base class has SupportedMediaTypes property
  - ContentTypes property or [Produces] filter can be set explicitly to limit formatters
  - If Accept contains "*/*" then rest of Accept values ignored
    - RespectBrowserAcceptHeader on MvcOptions can change behavior

Formatter	Accept type	Comment
StringOutputFormatter	text/plain	
JsonOutputFormatter	application/json, text/json	
XmlSerializerOutputFormatter	application/xml, text/xml	Not registered by default
XmlDataContractSerializerOutputFormatter	application/xml, text/xml	Not registered by default

FormatFilter & FormatFilterAttribute

- Allows overriding of Accept header
  - Looks for "format" route or query param
  - FormatterMappings on MvcOptions indicates format to media type mapping
- Sets ContentTypes on ObjectResult

```
public void Configure(IApplicationBuilder app)
{
    app.UseMvc(routes =>
    {
        routes.MapRoute("default",
            "api/{controller} ");

        routes.MapRoute("formatted",
            "api/{controller}.{format}");
    });
}
```

[Produces] & [Consumes] attributes

- Used to control what formatters used on request/response

```
[HttpGet]
[Consumes("application/xml")]
[Produces("application/json"")]
public object Get()
{
    return new {...};
}
```

XML compatibility shim

- Library to help migrate from Web API to Core MVC
  - Microsoft.AspNetCore.Mvc.WebApiCompatShim
- Provides old classes that map to the new framework
  - ApiController
  - FromUriAttribute
  - HttpRequestMessage helpers/extensions/model binder
  - HttpResponseMessage helpers/extensions/formatter
  - HttpResponseException
  - HttpError

Error handling

- HandleError from MVC 5 has been removed
- Resource filter's post processing runs after exception filters
  - Last chance place to "handle" exceptions with a result
  - Or just can log exceptions

Error pages

- Diagnostics middleware
  - Microsoft.AspNetCore.Diagnostics
  - UseDeveloperExceptionPage useful for development/debugging error info
  - UseExceptionHandler useful for production error pages
    - Logs error information
    - Invokes error path

Summary

- Brave new (yet somewhat familiar) world
- .NET Core is a cross-platform framework
- ASP.NET Core is a flexible HTTP pipeline architecture
- MVC and Web API have had a quite a make over