.NET Framework: Developing Modern Web Apps with ASP.NET MVC – Workshop*PLUS*

Wael Kdouh

Senior Consultant

v1.0

Module 1: Overview

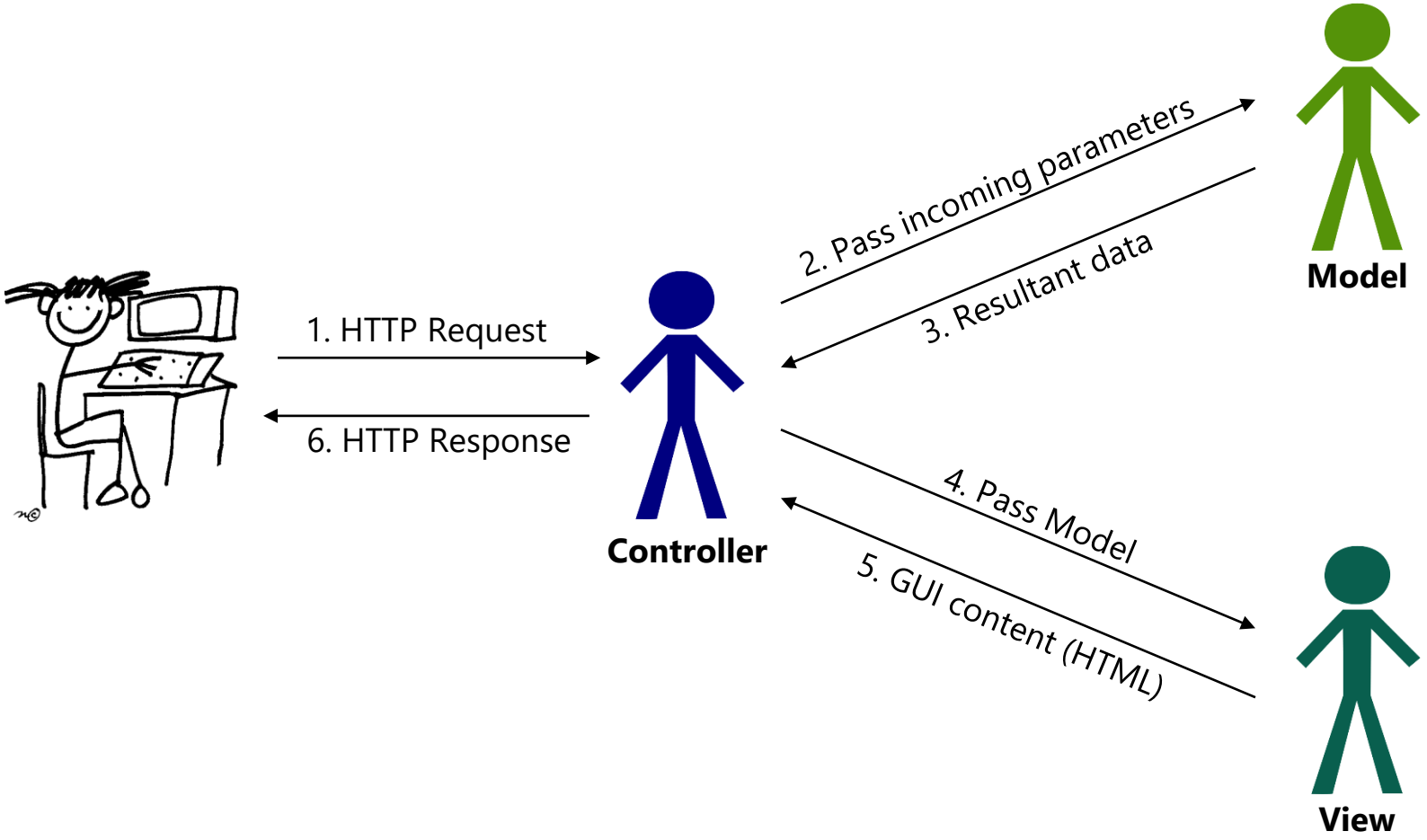
Module Overview

Module 1: Overview

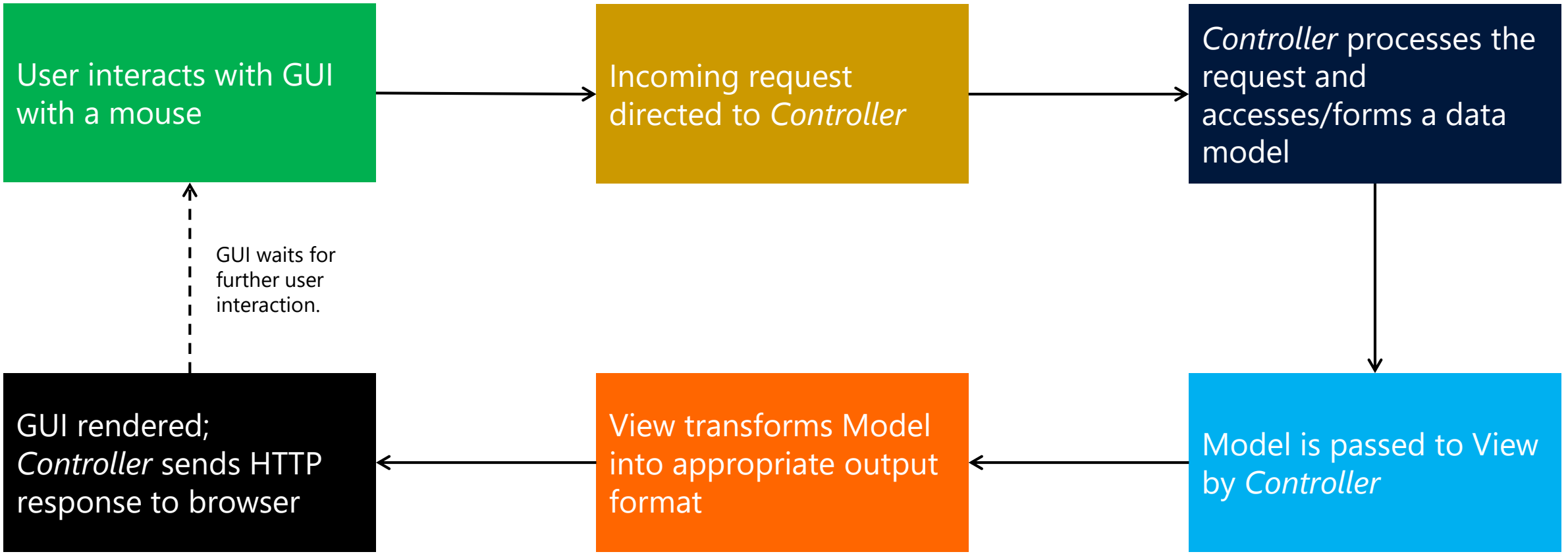
Section 1: Design Patterns

Lesson: MVC Design Pattern

Model View Controller (MVC) Design Pattern



MVC Control Flow



MVC Design Pattern

- Separation of Concerns (SoC)
- Search Engine Optimization (SEO) and Representational State Transfer (REST) friendly URL
- Test Driven Development (TDD)
- Better Integration with JavaScript
- Stateless design
- Full control over the rendered HTML
- Less code duplication; promotes maintainability

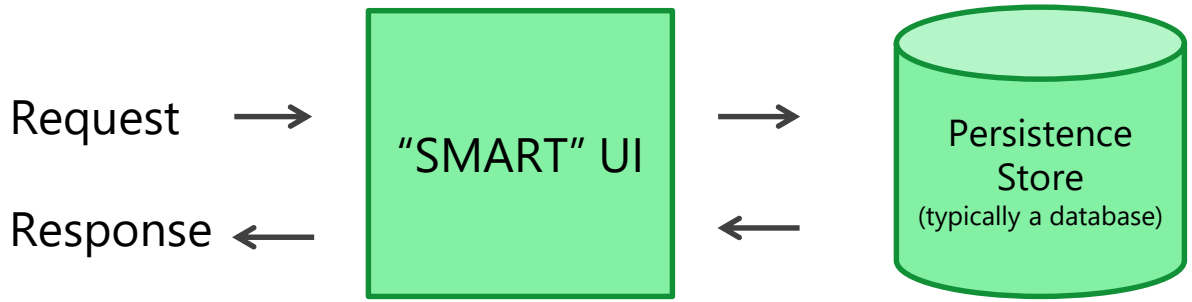
Module 1: Overview

Section 1: Design Patterns

Lesson: MVC vs. Web Forms vs. Web Pages

Web Forms

- Web Forms follow “Smart UI” Pattern (most common design pattern)
- Tightly-coupled and event driven
- UI and business logic is mixed together
- Fragmented and difficult to maintain and extend over time
- Difficult to fully Test



Web Forms Values

- Familiar control and event-based programming model
- Controls encapsulate HTML, JavaScript and Cascading Style Sheets (CSS)
- Support data binding and templating
- Rich UI controls included – datagrids, charts, Asynchronous JavaScript And XML (AJAX)
  - Help achieve common tasks
- Browser differences handles for you
  - Browser differences are handled by controls

SharePoint is an example of an application built using Web Forms

Web Pages Values

- Programming model built around individual pages
  - Similar to Classic ASP or PHP
- Easy to pick up and learn
- Inline scripting model with Razor and C#/Microsoft Visual Basic .NET
  - Full power of Microsoft .NET Framework is available
- Simplified model with top-to-bottom execution
- Full control over HTML
- No page lifecycle
  - No Page Load, Page Render, etc.

WebMatrix is an excellent lightweight IDE for quick Web Page development

MVC Values

- Slightly lower level programming model
- No higher-level abstraction by controls
- Requires deeper understanding of HTML, JS
  - Feels comfortable for many traditional web developers
- Total control of HTML markup, JS, CSS
- Supports unit testing, TDD, and agile methodologies
- Encourage more prescriptive applications
- Extremely flexible and extensible

Module 1: Overview

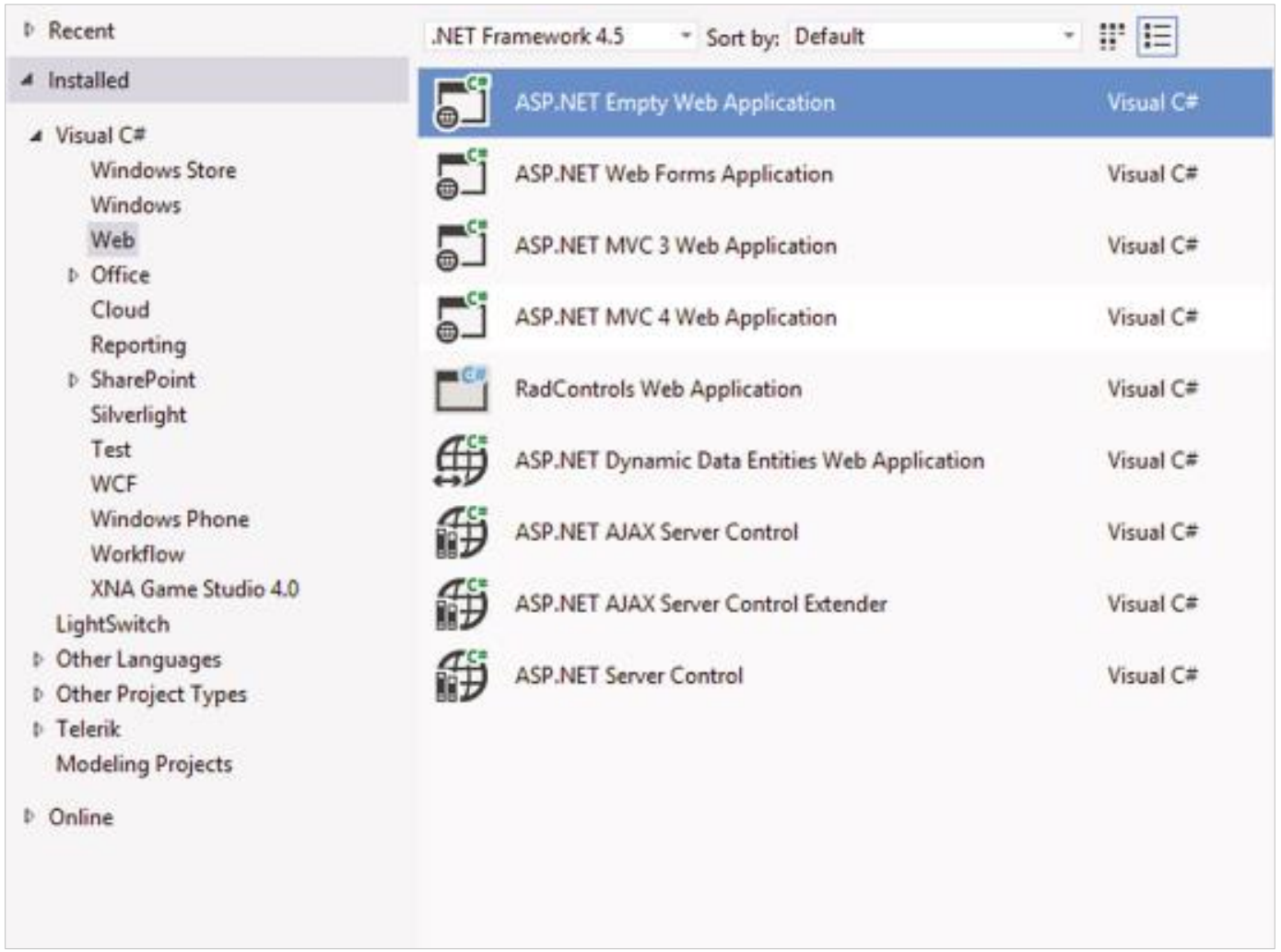
Section 2: ASP.NET

Lesson: One ASP.NET – MVC, Web Forms, and Web Pages

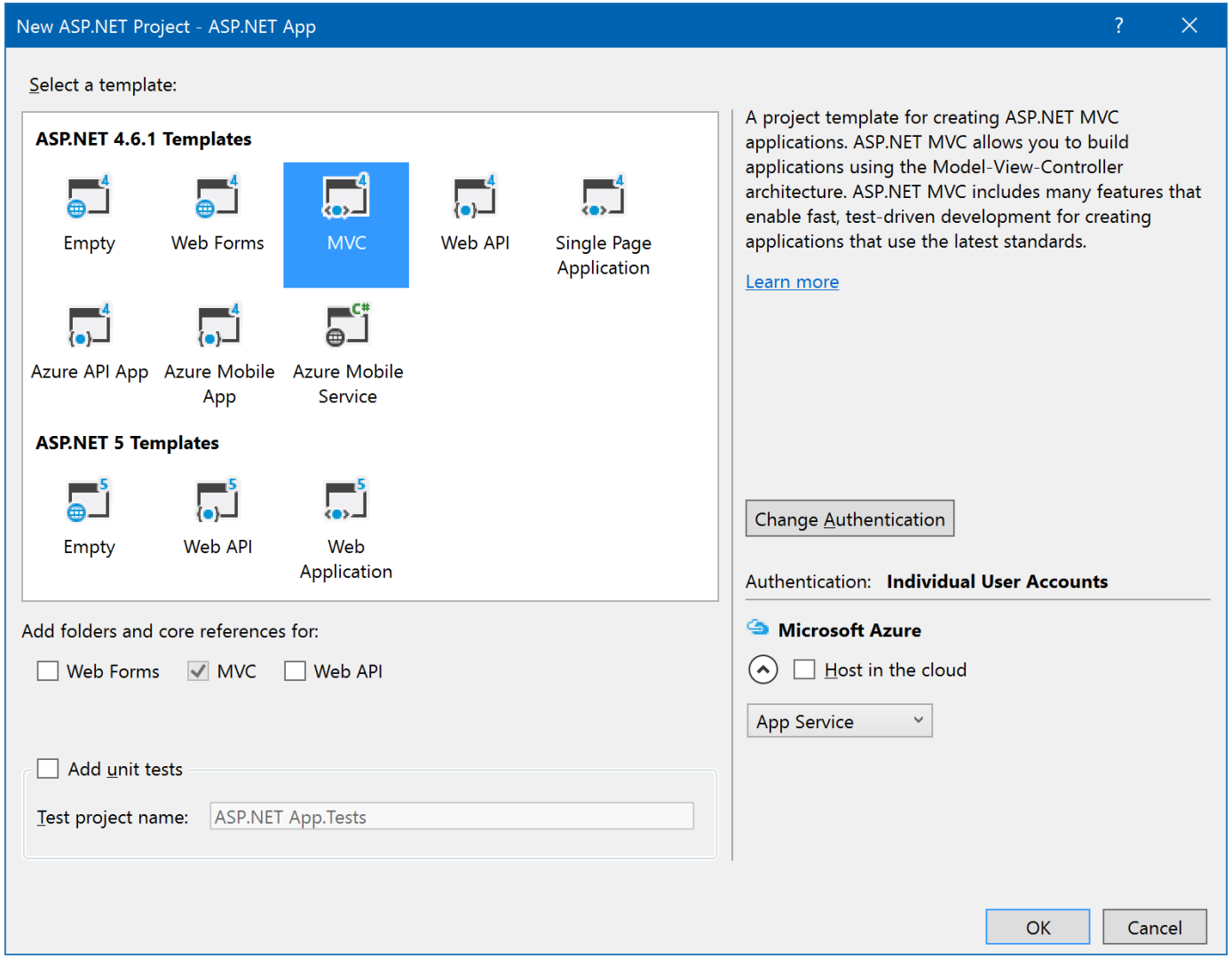
Earlier, you had to choose one model over another

Beginning with Visual Studio 2013, it is one ASP.NET

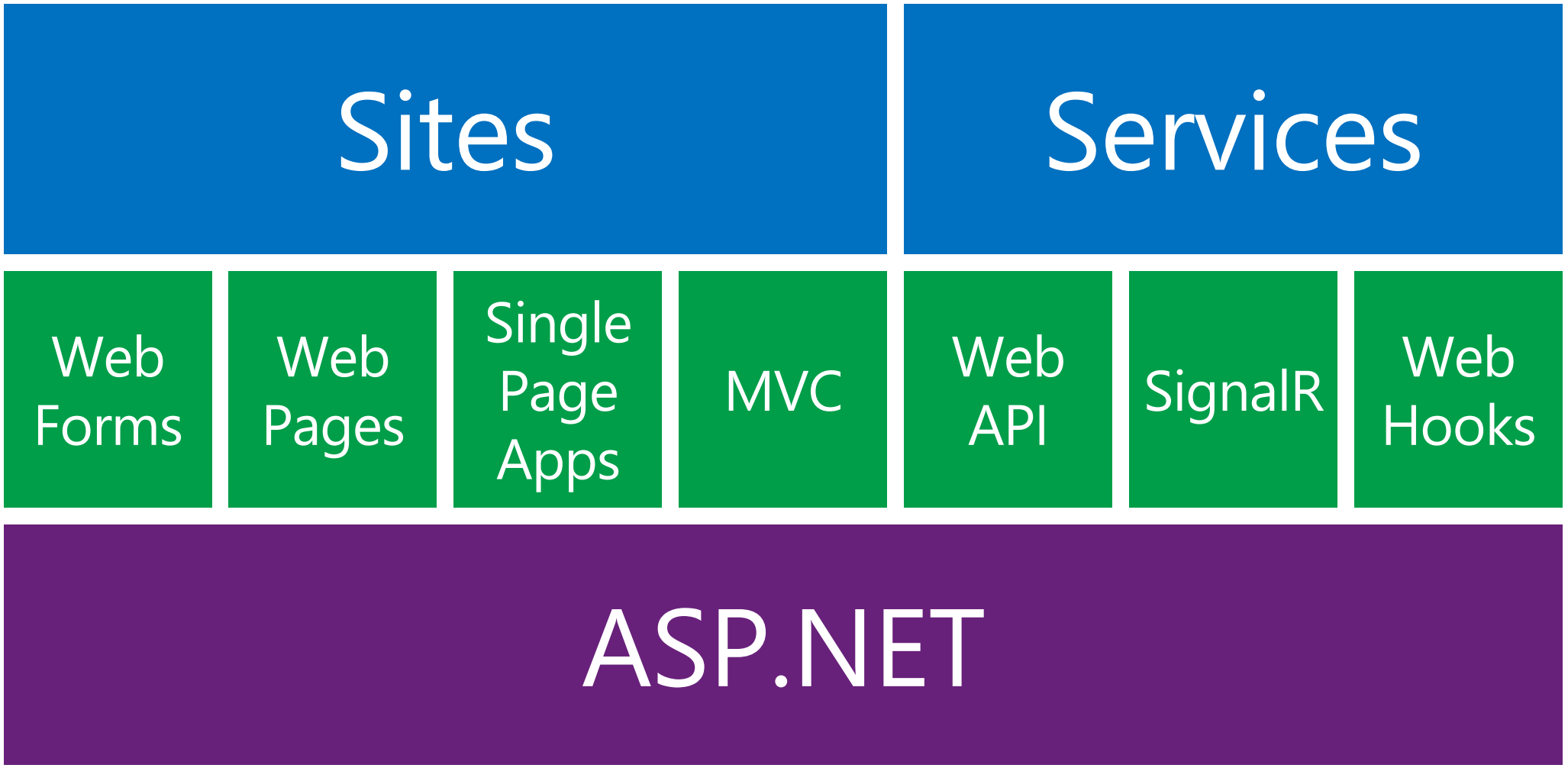
Visual Studio 2012 Project System



One ASP.NET Project System



One ASP.NET


Sites
Services
Web Forms
Web Pages
Single Page Apps
MVC
Web API
SignalR
Web Hooks
ASP.NET

Commonalities

- All programming models have the same Microsoft ASP.NET
  - Authentication/Authorization/Membership
  - Output Caching, Session State, and Configuration
  - AJAX, Deployment, etc.
- All programming models are fully supported and will continue to be so
- All programming models solve real problems

Demo: One ASP.NET

Module 1: Overview

Section 2: ASP.NET

Lesson: Web API, SignalR, and SPA

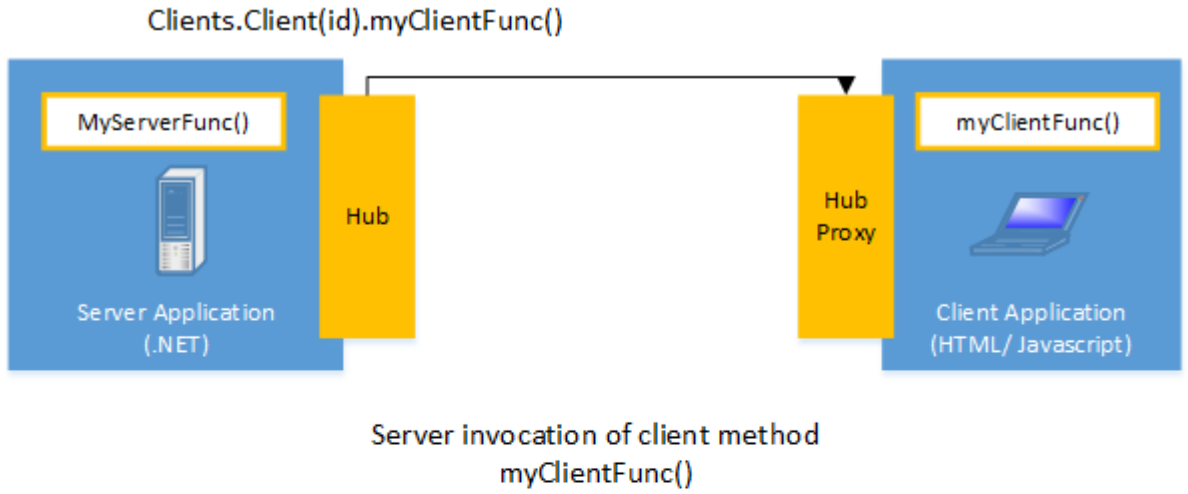
ASP.NET Web API

- Framework for building and consuming HTTP services
- Reachable by a broad range of clients including browsers and mobile devices
  - A browser is all you need!
- Build RESTful services

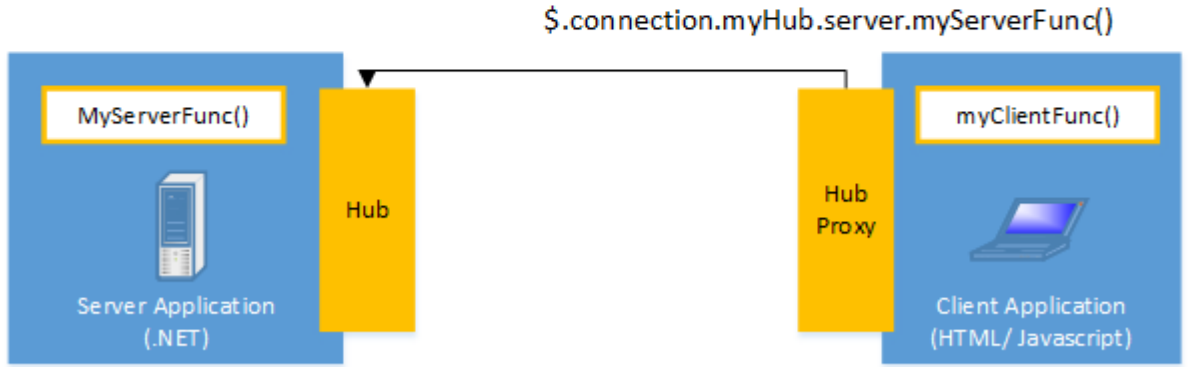
ASP.NET SignalR

- ASP.NET library to add real-time web functionality
- Real-time Web: Ability to have server code push content to connected clients instantly as it becomes available
- SignalR API enables server side code (C#) to call JavaScript functions in client browser through RPC calls

ASP.NET SignalR



Server invocation of client method
myClientFunc()



Client invocation of server method
MyServerFunc()

Single Page Application (SPA)

- Entire page is loaded in browser after the initial request
- Subsequent requests take place through AJAX
- Technologies used:
  - Server-side Development: REST-based services (for example, **ASP.NET Web API**)
  - Client-side Development: JavaScript framework (for example, **AngularJS**)
  - Client-side Styling: Cascading Style Sheets (CSS) framework (for example, **Bootstrap**)

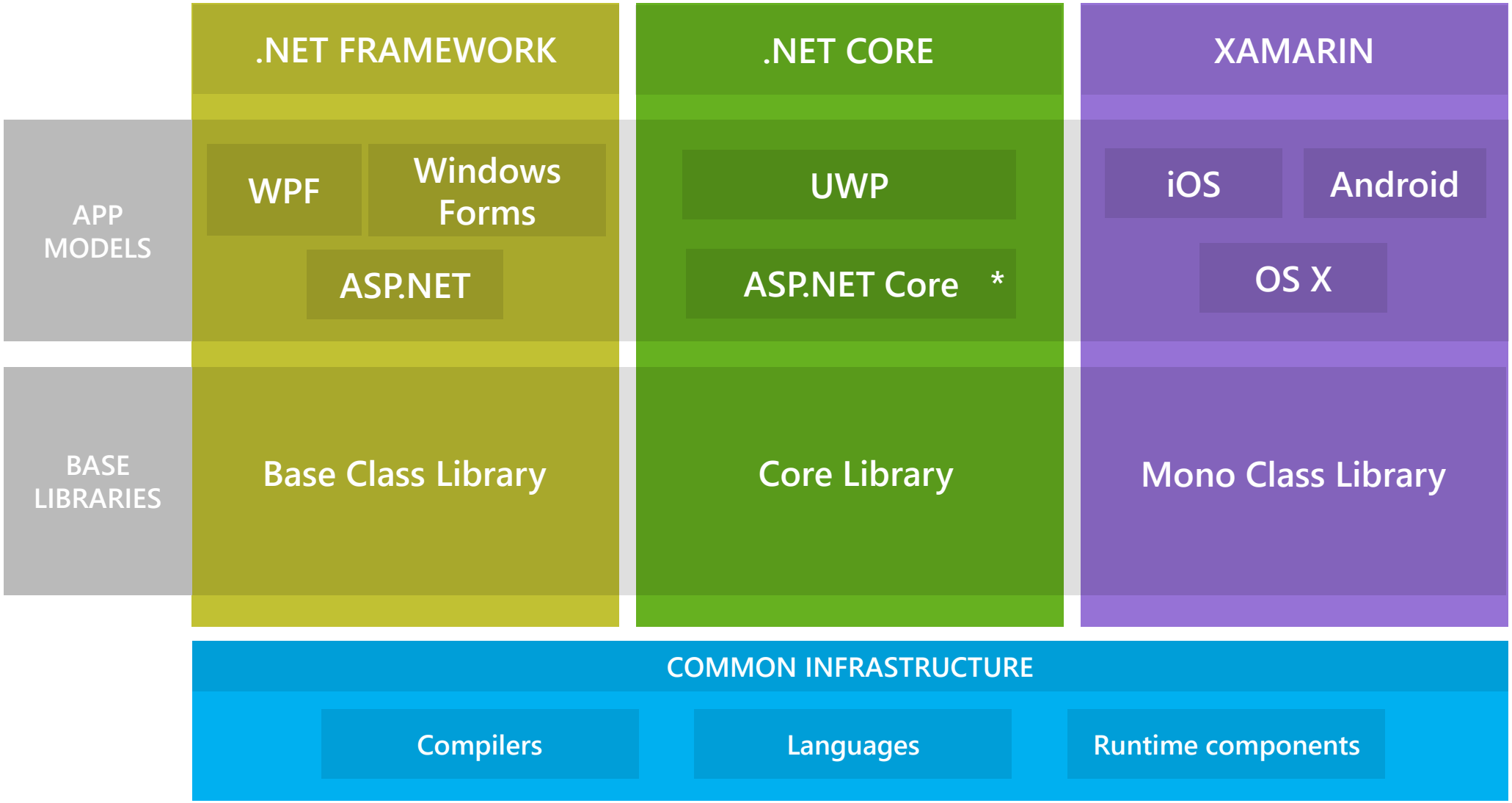
Demo: ASP.NET SignalR

Module 1: Overview

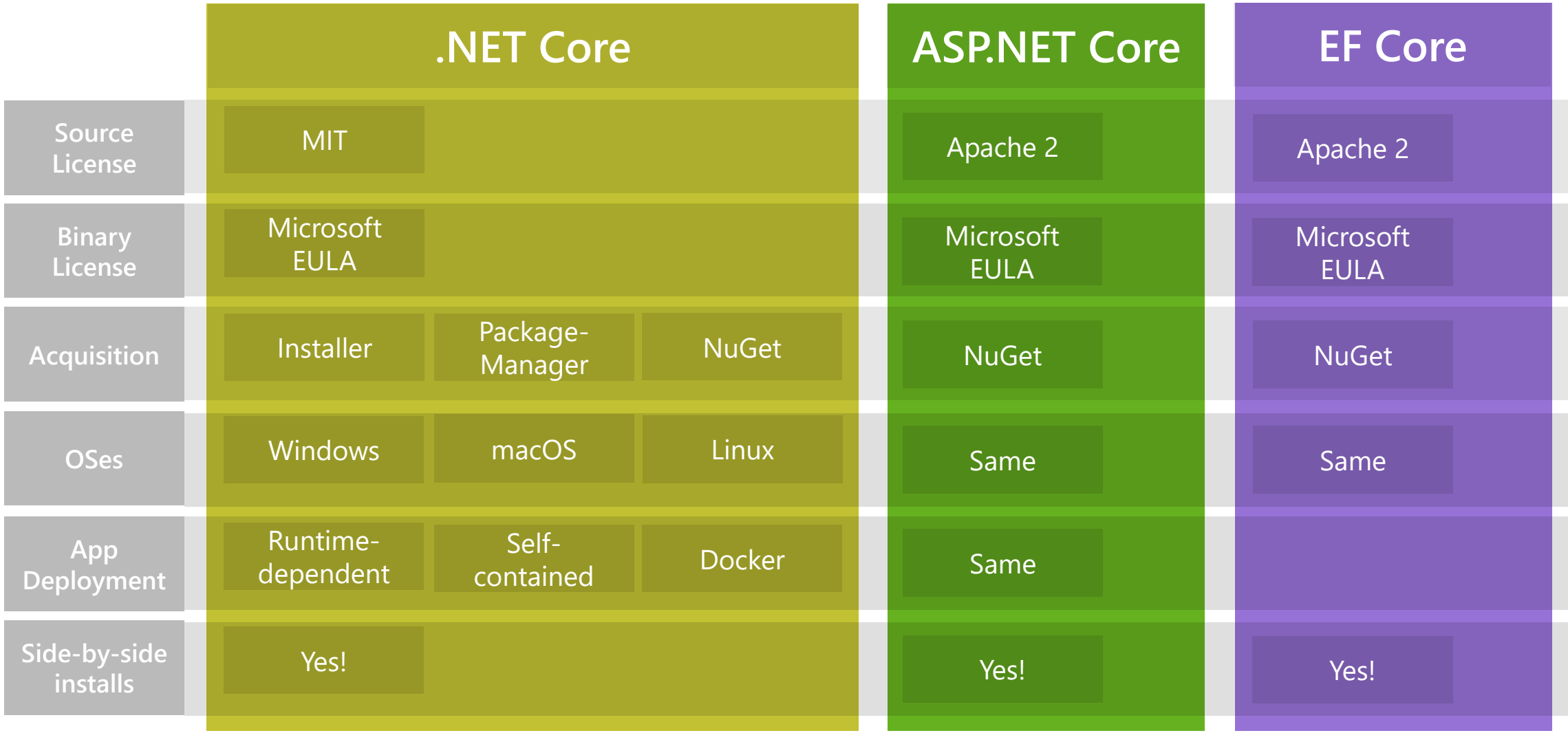
Section 3: .NET Platform

Lesson: Overview

Open .NET Ecosystem



.NET Core


.NET Core
ASP.NET Core
EF Core
Source License
MIT
Apache 2
Apache 2
Binary License
Microsoft EULA
Microsoft EULA
Microsoft EULA
Acquisition
Installer
Package-Manager
NuGet
NuGet
NuGet
OSes
Windows
macOS
Linux
Same
Same
App Deployment
Runtime-dependent
Self-contained
Docker
Same
Side-by-side installs
Yes!
Yes!
Yes!

.NET Foundation

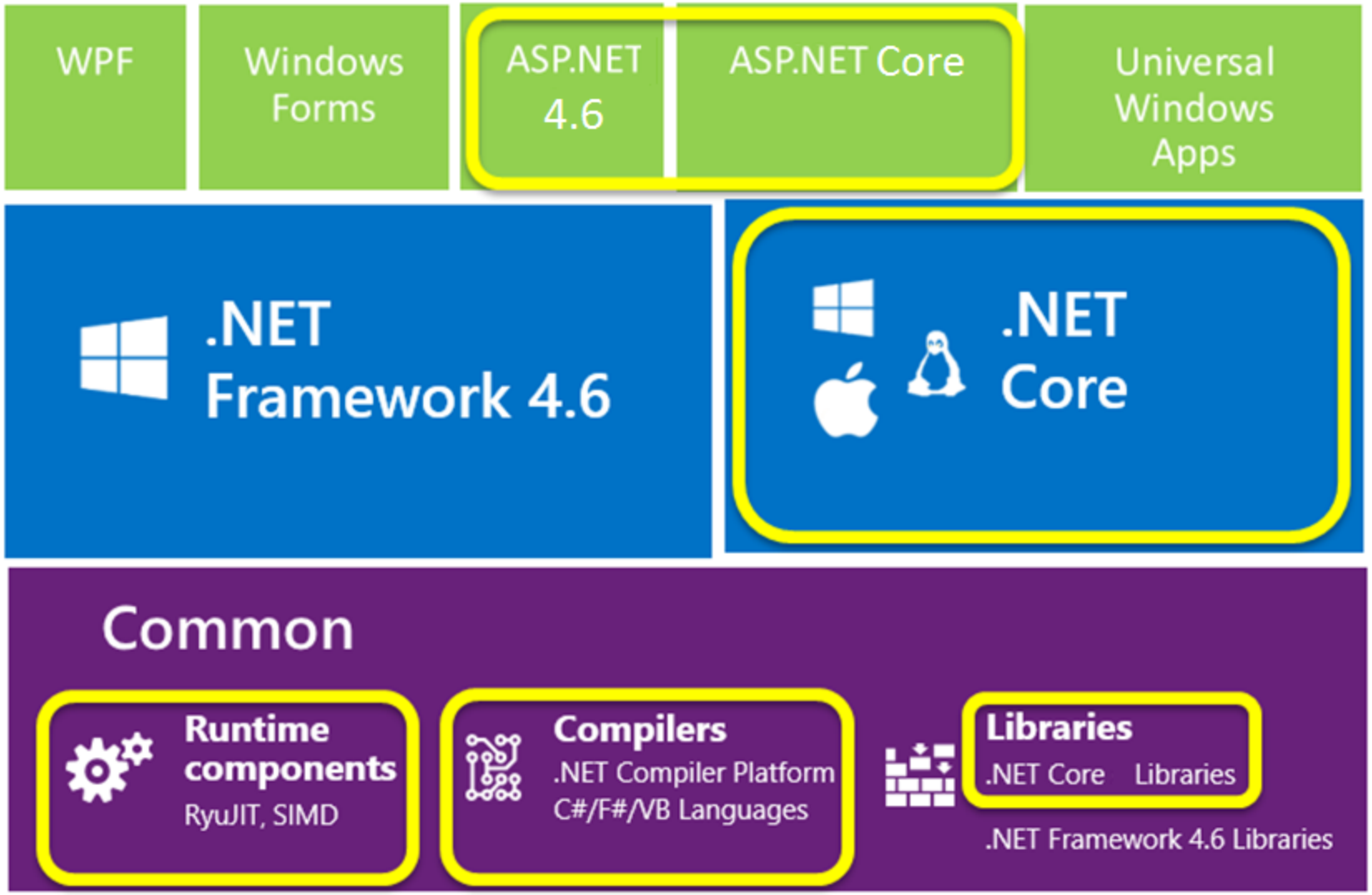
Openness

Community

Rapid innovation

.NET Compiler Platform ("Roslyn")
MVVM Light Toolkit
MSBuild
ASP.NET MVC
IdentityManager
MEF
.NET SDK for Hadoop
ASP.NET Core
.NET Core
Kudu
.NET Micro Framework
Mailkit
Cecil
Xamarin.Auth
Mimekit
Xamarin SDK
Umbraco
Open Live Writer
NuGet
WorldWide Telescope
Couchbase Lite for .NET
ASP.NET Web Pages
ASP.NET SignalR
Microsoft Azure SDK for .NET
WCF
LLILC
Mono
Entity Framework
Open XML SDK
IdentityServer
Microsoft Azure WebJobs SDK
Microsoft Web Protection Library
OWIN Authentication Middleware
ProtoBuild
ASP.NET Web API
Orchard CMS
System.Drawing
Orleans
ASP.NET AJAX Control Toolkit
Prism
Xamarin.Mobile
Salesforce Toolkits for .NET

What Is Open Right Now?



ASP.NET vs. ASP.NET Core

MSBuild/CodeDOM > csc.exe	**Compilation**	.Net CLI (Roslyn)
Loose, GAC, NuGet	**Libraries**	NuGet, npm, Bower
FCL, GAC, NuGet	**Application Frameworks**	NuGet
IIS	**Web Server**	IIS, HTTP.SYS, Kestrel
.NET BCL and FCL	**Platform Libraries**	.NET BCL and FCL; .NET on NuGet
.NET CLR	**Runtime**	.NET CLR; .NET Core CLR
IIS: WebEngine4.dll; EXE: OS	**Runtime Loader**	.Net CLI
Windows	**Operating System**	Windows, OSX, Linux

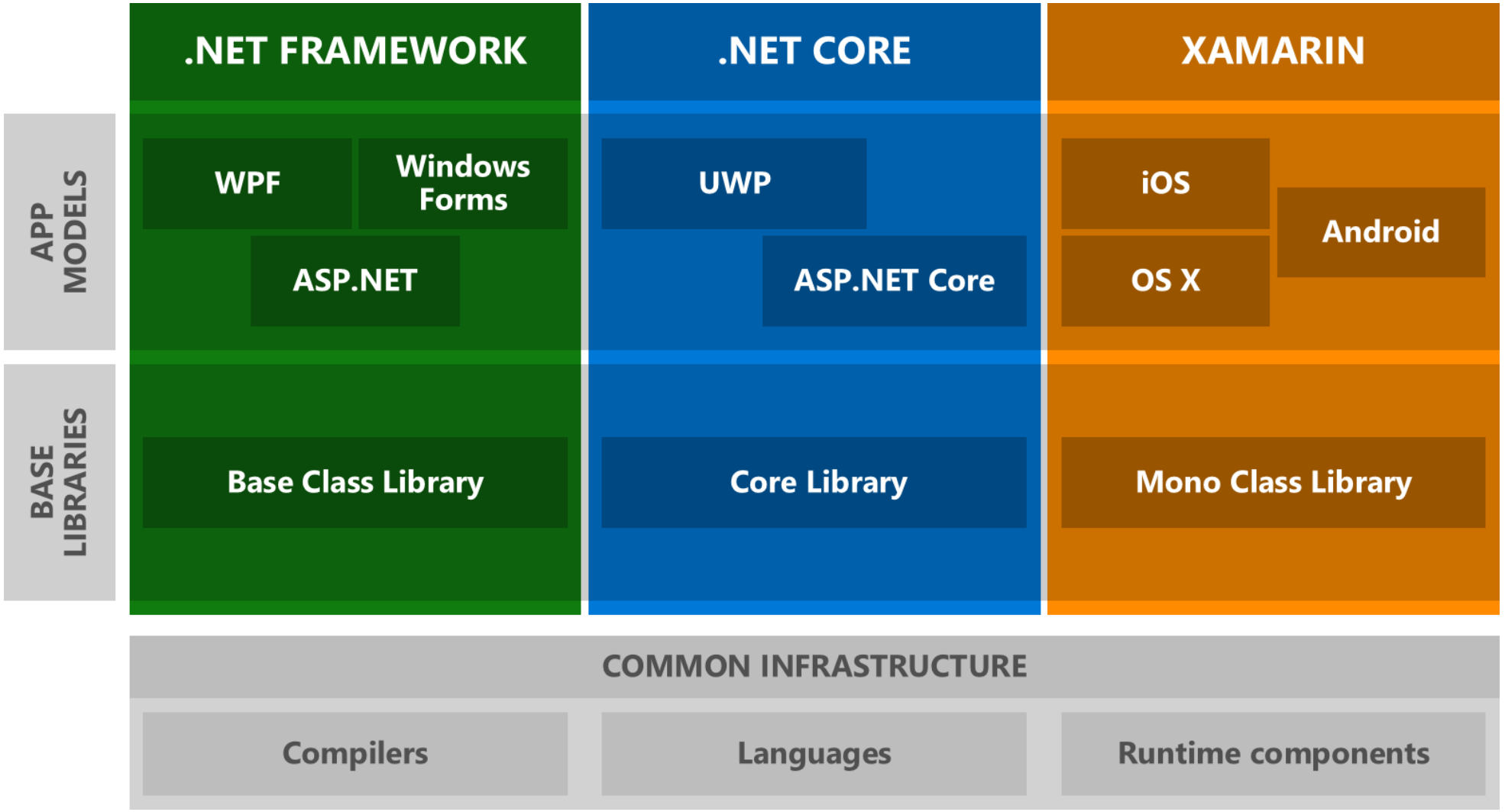
.NET Framework vs. .NET Core (Server Apps)

.NET Framework	.NET Core
Current application runs on .NET framework. Recommended to extend it instead of migrating.	Cross-platform needs
Need 3rd party libraries not available on .NET Core	Targeting microservices
Need.NET technologies not available on .NET Core	Using Docker containers
Need a platform not supported by .NET Core	Need high performance & scalable systems
	Side-by-side .NET versions by application
	Fully open-source

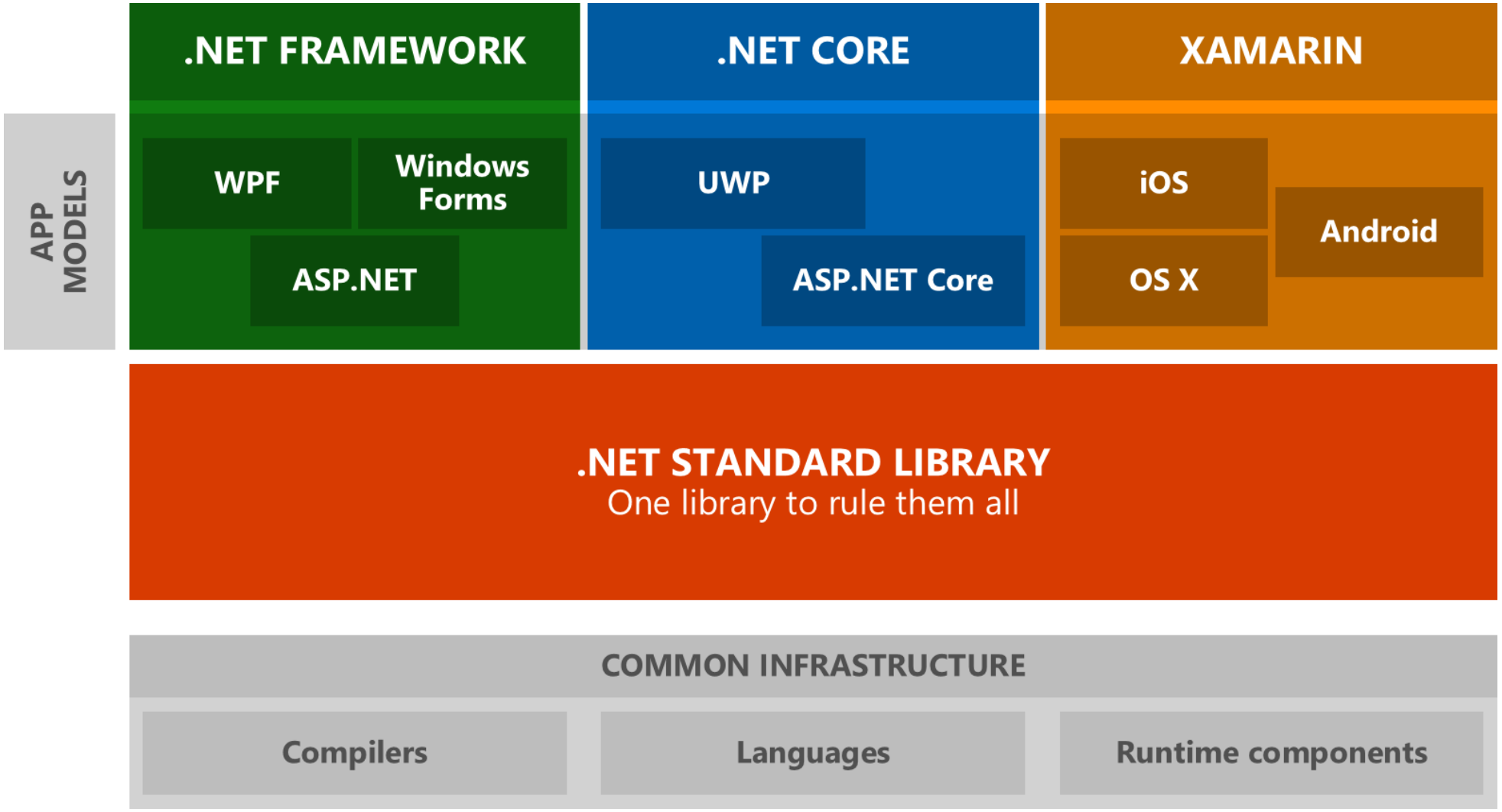
.NET Standard Library

- Goal: Establish greater uniformity in the .NET ecosystem
- A set of APIs that all .NET platforms have to implement
- Unifies the .NET platform and prevents future fragmentation
- .NET Standard will replace Portable Class Libraries (PCLs)
- Addresses three main scenarios:
  - Defines uniform set of BCL APIs for all .NET platforms to implement, independent of workload
  - Enables developers to produce portable libraries that are usable across .NET runtimes, using the same set of APIs
  - Reduces and hopefully eliminates conditional compilation of shared source due to .NET APIs

.NET with Framework Base Libraries

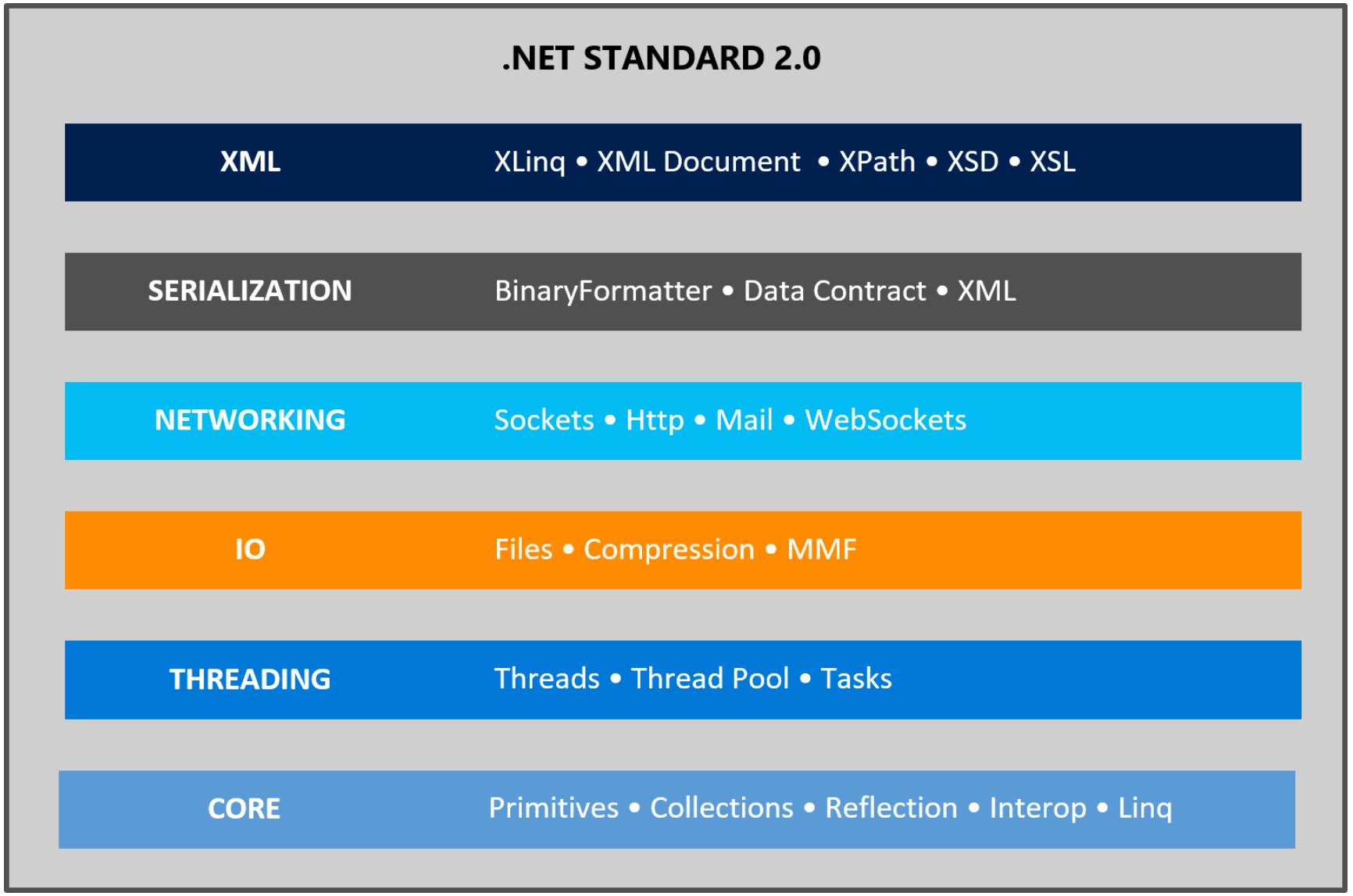


.NET Standard Library



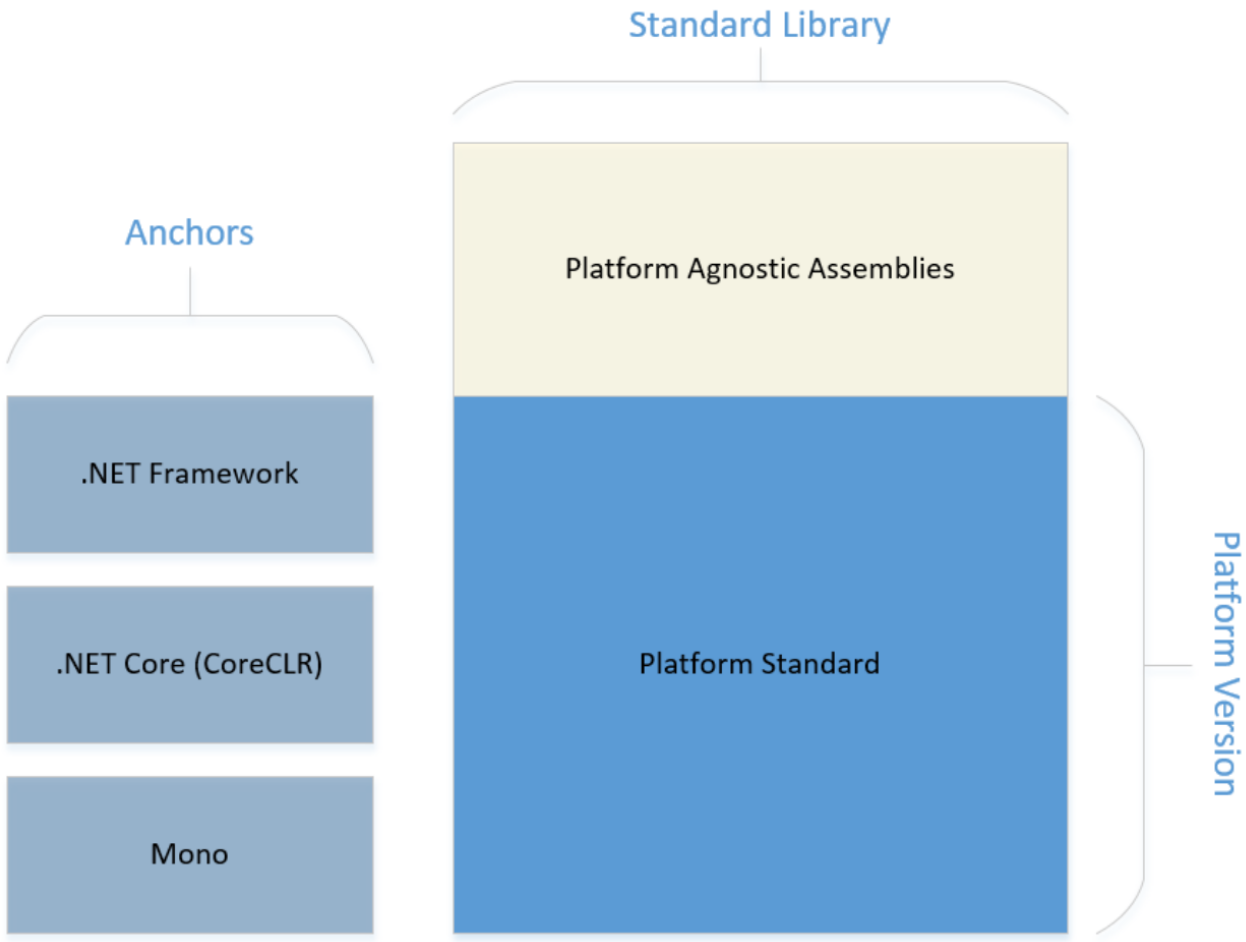
.NET Standard Library

.NET Platform	.NET Standard							
	1.0	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**2.0**
.NET Core	→	→	→	→	→	→	1.0	vNext
.NET Framework	→	4.5	4.5.1	4.6	4.6.1	4.6.2	vNext	4.6.1
Xamarin.iOS	→	→	→	→	→	→	→	vNext
Xamarin.Android	→	→	→	→	→	→	→	vNext
Universal Windows Platform	→	→	→	→	10.0	→	→	vNext
Windows	→	8.0	8.1					
Windows Phone	→	→	8.1					
Windows Phone Silverlight	8.0							



* The above API footprint is subject to change. APIs may be added/removed. It's accurate as of Jan 2017.

Mapping the .NET Standard Library to Platforms



PLATFORM NAME	ALIAS							
.NET Standard	netstandard	1.0	1.1	1.2	1.3	1.4	1.5	1.6
.NET Core	netcoreapp	→	→	→	→	→	→	1.0
.NET Framework	net	→	4.5	4.5.1	4.6	4.6.1	4.6.2	vNext
Mono/Xamarin Platforms		→	→	→	→	→	→	*
Universal Windows Platform	uap	→	→	→	→	10.0		
Windows	win	→	8.0	8.1				
Windows Phone	wpa	→	→	8.1				
Windows Phone Silverlight	wp	8.0						

Demo: .Net Standard

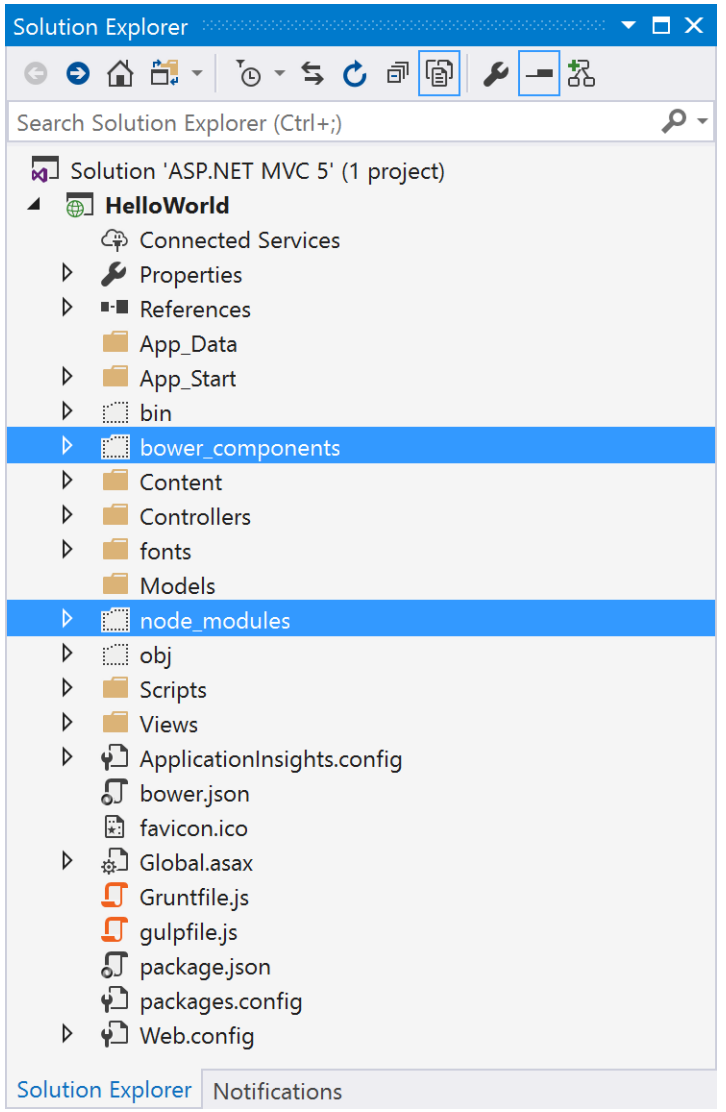
Module 1: Overview

Section 3: ASP.NET MVC 5

Lesson: ASP.NET MVC 5 Projects

ASP.NET MVC Project Layout

- Project files
  - .csproj
  - .vbproj
- Support for:
  - NuGet
  - NPM
  - Bower



Package Managers

- NuGet
  - Package manager for Microsoft development platform
  - Great for server-side libraries: more than 55,000 packages available
  - .NET Framework and runtime now available through NuGet
- NPM
  - Package manager for JavaScript; widely used in JavaScript development community
  - First class citizen in ASP.NET and Visual Studio 2015
  - More than 250,000 packages are available
- Bower
  - Package manager for the web (HTML, JavaScript, and CSS)
  - Installed using NPM; suited for web application front-end development
  - More than 36,000 packages are available

Task Runners

- Tasks runners are used to achieve automation for client-side code:
  - LESS/SL compilation to CSS
  - JavaScript minification
  - JSLint/JSHint
  - JavaScript unit tests
- Grunt
  - Most popular task runner for JavaScript
  - More than 5800+ plug-ins are available
- Gulp
  - Task runner for JavaScript
  - Relies on the code logic based on the pipes for simplification
  - More than 2600+ plug-ins

Demo:

1. ASP.NET MVC 5 Project and Configuration System

Module 1: Overview

Section 4: ASP.NET MVC

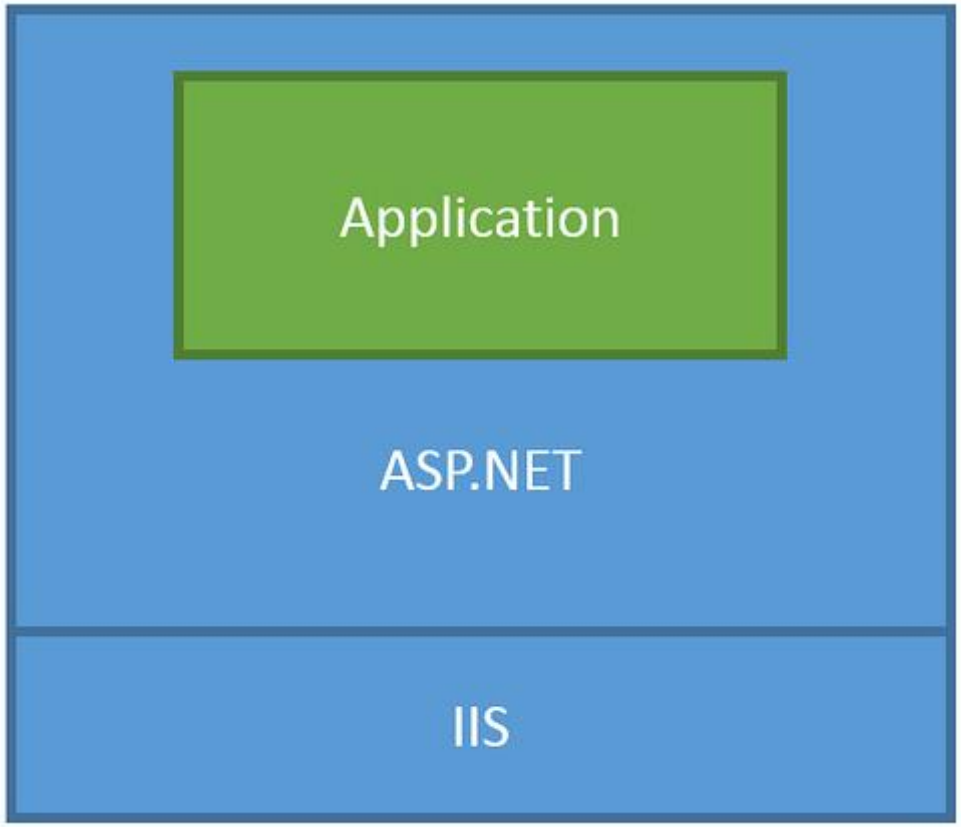
Lesson: Middleware

Middleware

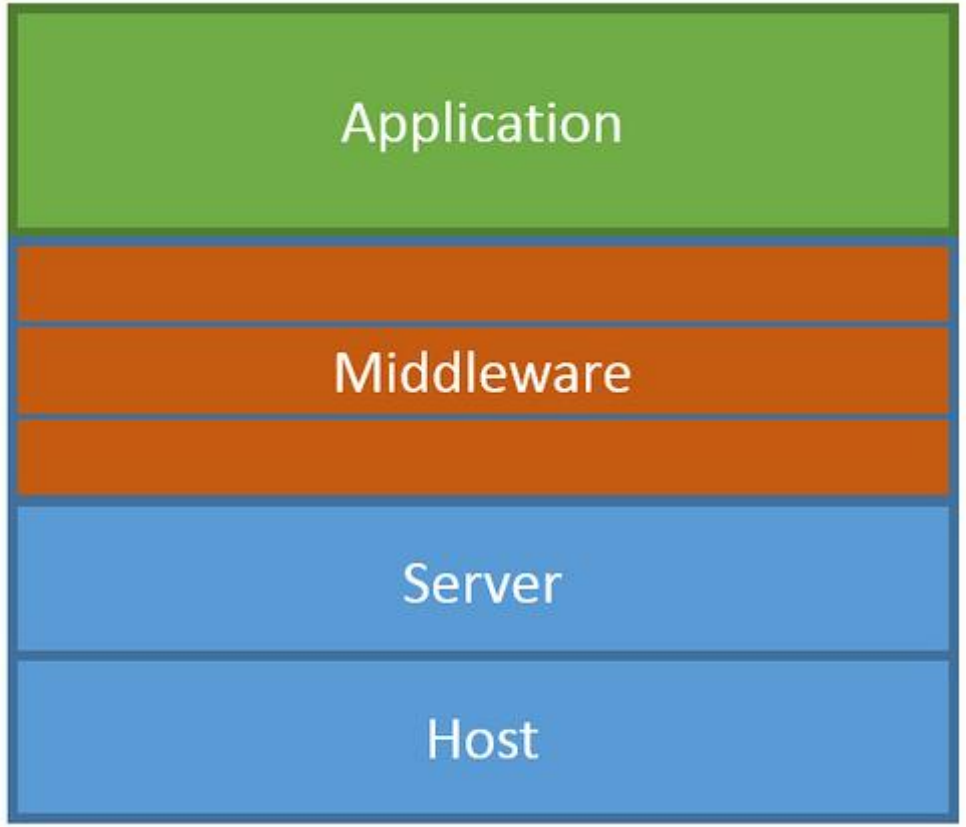
- Small application components assembled into an application pipeline to handle requests and responses
- Integrated support by ASP.NET MVC 5
- Wired up in **Configure** method of **Startup** class
- Either invokes the next component in chain or short-circuits it
- **Run**, **Map**, and **Use** extension methods
  - MVC engine is configured in Request Pipeline through Use extension method
- Implemented in-line as anonymous method or through reusable class
- Order of **Use[Middleware]** statements in application's Configure method is very important

ASP.NET MVC 5 Middleware

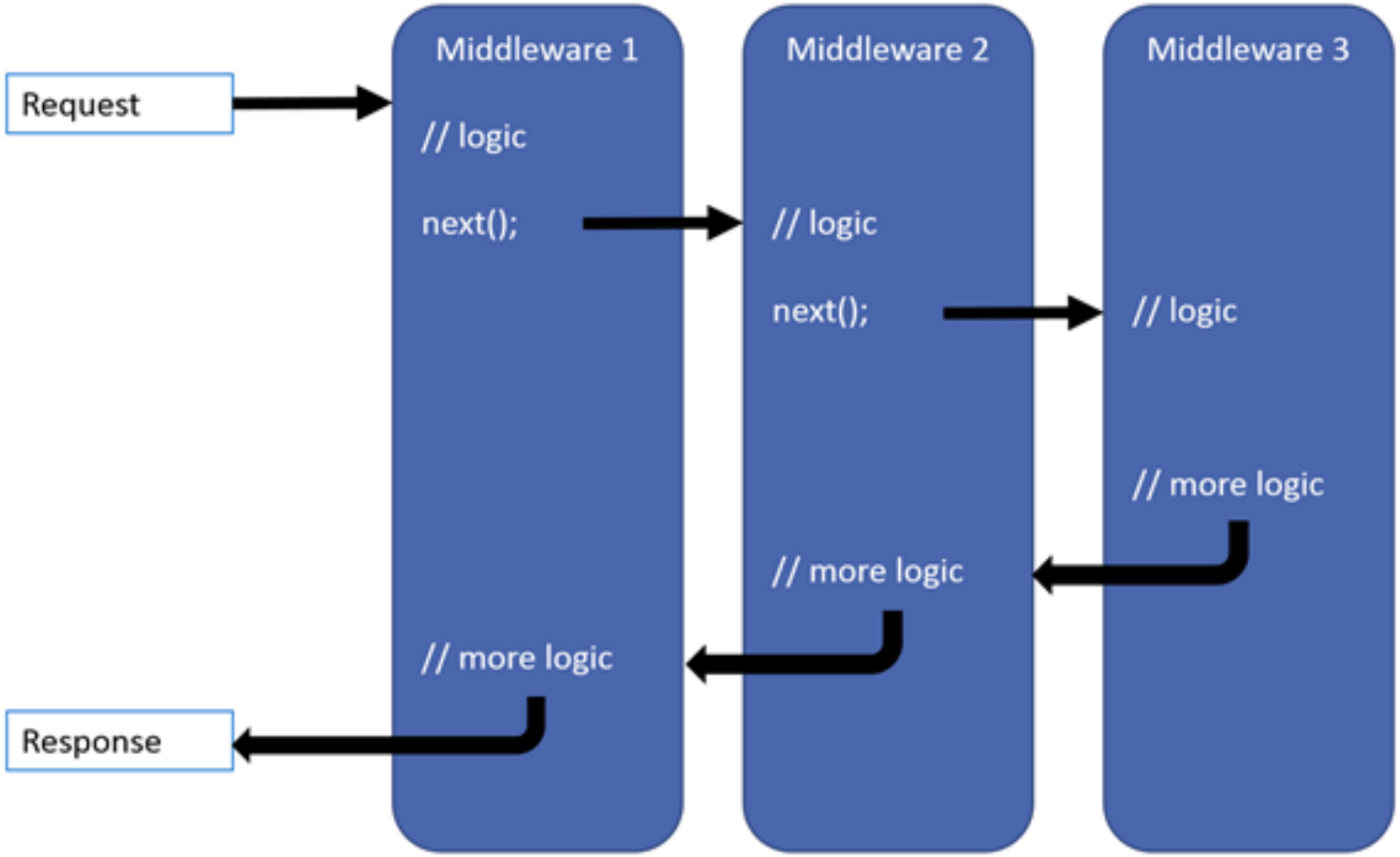
Traditional ASP.NET Application Model



ASP.NET MVC 5 Middleware



Middleware Pipeline



ASP.NET MVC Request Pipeline

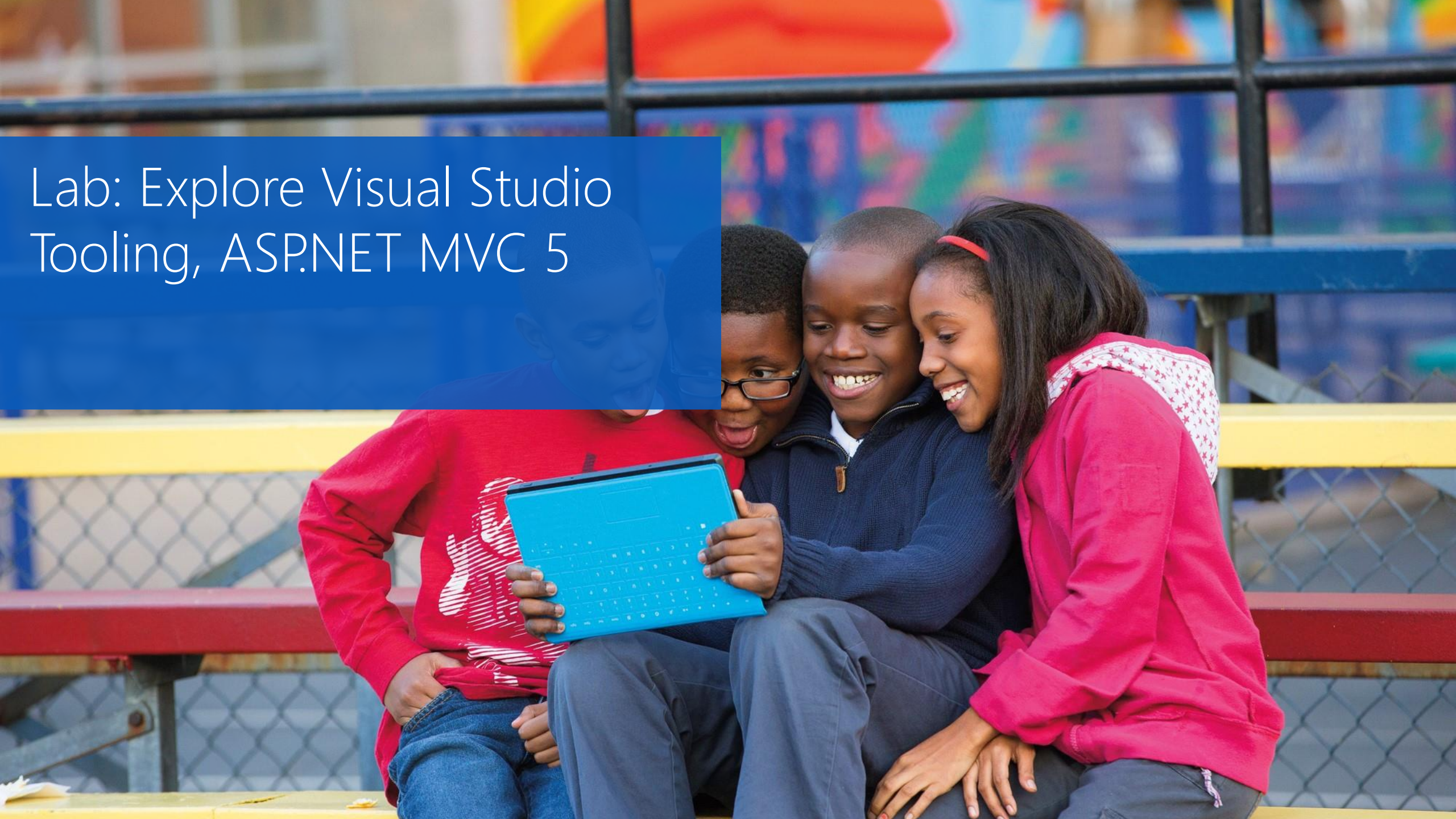
```
app.UseCookieAuthentication(new CookieAuthenticationOptions
{
    AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
    LoginPath = new PathString("/Account/Login"),
    Provider = new CookieAuthenticationProvider
    {
        // Enables the application to validate the security stamp when the user logs in.
        // This is a security feature which is used when you change a password or add an external lo
        OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, Appli
            validateInterval: TimeSpan.FromMinutes(30),
            regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))
    }
});
app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

// Enables the application to temporarily store user information when they are verifying the second
app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie, TimeSpan.FromMinutes(5));

// Enables the application to remember the second login verification factor such as phone or email.
// Once you check this option, your second step of verification during the login process will be rem
// This is similar to the RememberMe option when you log in.
app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);

// Uncomment the following lines to enable logging in with third party login providers
app.UseMicrosoftAccountAuthentication(
    clientId: "",
    clientSecret: "");

app.UseTwitterAuthentication(
   consumerKey: "",
   consumerSecret: "");

app.UseFacebookAuthentication(
   appId: "",
   appSecret: "");

app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
```

Demo: Writing Middleware

Module Summary

- In this module, you learnt the following:
  - MVC Design Pattern
  - .NET Platform: .NET Framework & .NET Core
  - ASP.NET MVC 5 and Modern Web
  - ASP.NET MVC 5 Project System
  - .NET Framework versus .NET Core
  - .NET Platform Standard
  - Middleware

Lab: Explore Visual Studio Tooling, ASP.NET MVC 5




Microsoft