.NET Core & ASP.NET Core

22. april 2018

Agenda

Brief history of .NET Core

Birds eye view of .NET Core

Cross Platform Story

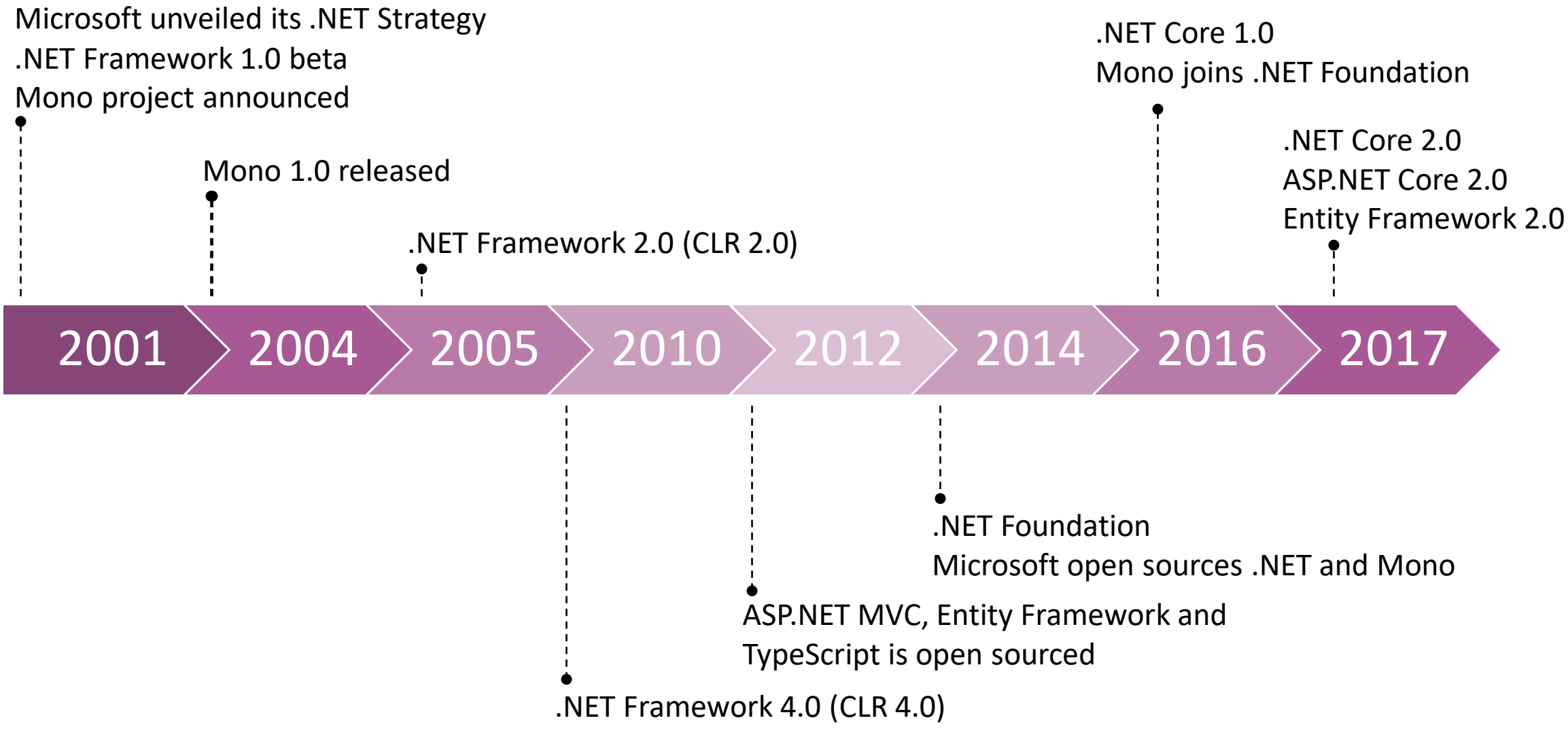
Brief history of ASP.NET Core

Working with ASP.NET Core in Visual Studio

Links to learn more

.NET Core

History of .NET Core



Features / Highlights

Modular and smaller implementation of .NET

Cross platform

App-level isolation

Open Source

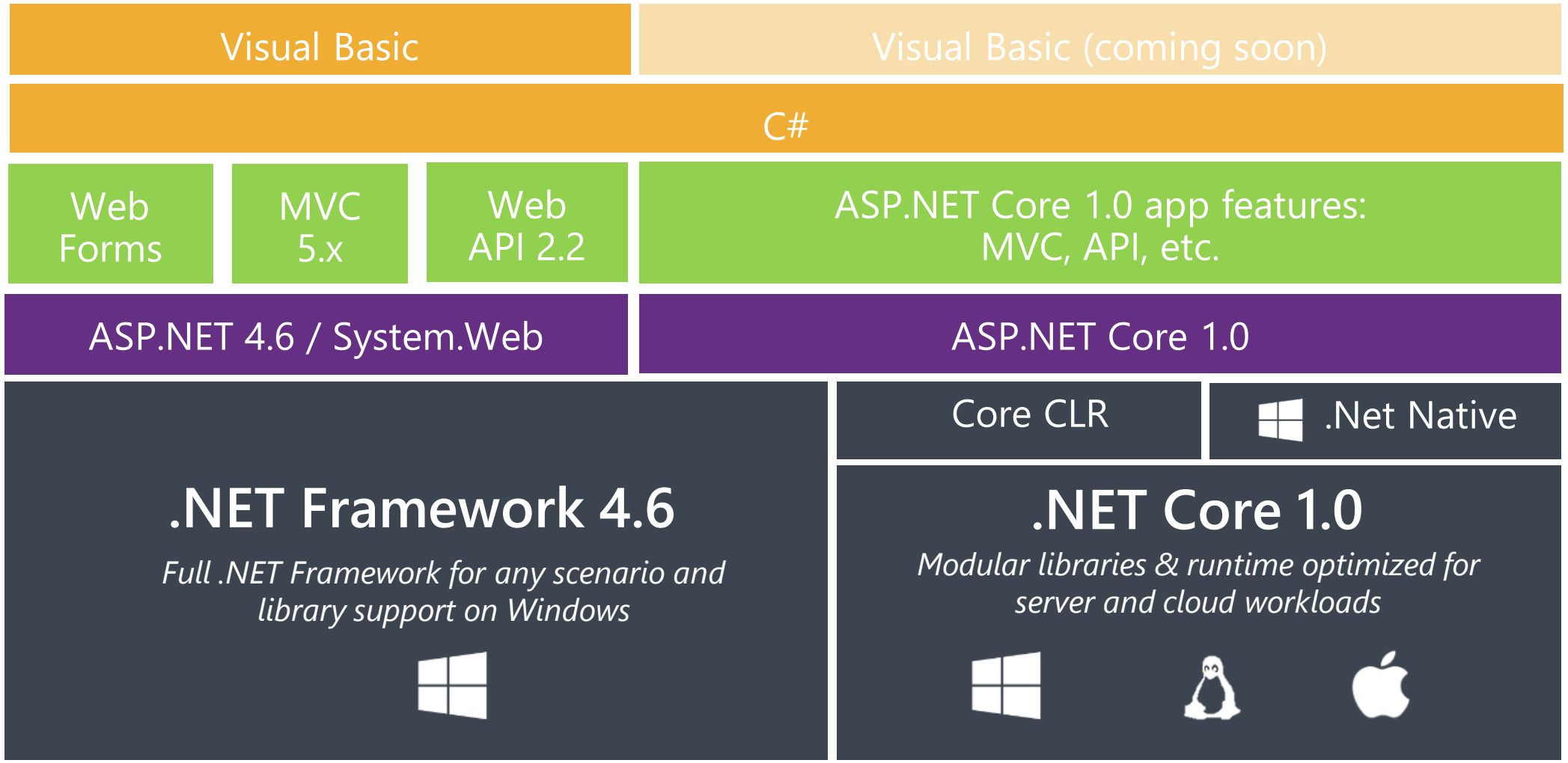
Optimized for specific workloads

Built for the coming decade of software development

Microservice architecture

Containers

.NET in a nutshell (2016)



What is not supported

- WPF
- WinForms
- WebForms
- Binary serialization
- Anything windows specific (Why is that?)
  - Registry, ACLs, perf counters, etc.

When to use .NET Core

Never

- WPF, WinForms
- ASP.NET WebForms

Always

- General purpose libraries

Maybe

- ASP.NET MVC
- Micro services
- Console applications

.NET Framework vs. .NET Core

.NET Core	.NET Framework
You need training to use	Easier for legacy teams
Windows, macOS and Linux on AMD64, x86 and ARM	Windows only on AMD64 and x86
Modular	A complete framework (All or nothing)
UWP, ASP.NET Core. Razor Pages, CLI	WPF, Windows Forms, ASP.NET Webforms or MVC
.NET Core is much more performant	Much less performant
You can use containers	Much harder to containerize
You're less dependent on 3rd party libraries for new features	Tight dependencies upon 3rd party libraries for newer features

.NET Framework vs. .NET Core (cont.)

.NET Framework	.NET Core	XAMARIN
WPF, Windows Forms, ASP.NET	UWP and ASP.NET Core	iOS, Android, OS X

.NET Standard
On library to rule them all
(Technically just a standard)

Common infrastructure

Compilers | Languages | Runtime components

ASP.NET Core

Overview

ASP.NET Core is a new open source and cross platform framework for building connected applications such as:

- Web applications
- Internet of Things (IoT) applications
- Mobile back-ends

ASP.NET Core runs on both .NET Framework 4.6 and .NET Core 2.0

.NET Core 2.0 Templates

- Console application
- Class library
- Unit Test project
- xUnit Test project
- ASP.NET Core Empty
- ASP.NET Core Web App MVC
- ASP.NET Core Razor Pages
- ASP.NET Core with Angular SPA
- ASP.NET Core with React.js SPA
- ASP.NET Core with React,js and Redux SPA
- ASP.NET Core Web API
- Community made templates are also available

ASP.NET Core

ASP.NET Core is no longer based on the old System.Web

Instead it is made up of a set of granular NuGet packages

This brings some benefits:

- Tighter security
- Reduced servicing
- Improved performance
- Decreased costs
  - Pay for what you actually use

The simple anatomy

Everything starts from the Program.cs Main method

All ASP.NET Core applications require a Starup class

Characteristic of modern web

Web Frameworks

- Responsive design
- Client frameworks
- Cloud ready
- Cross platform
- Open source

Web tooling

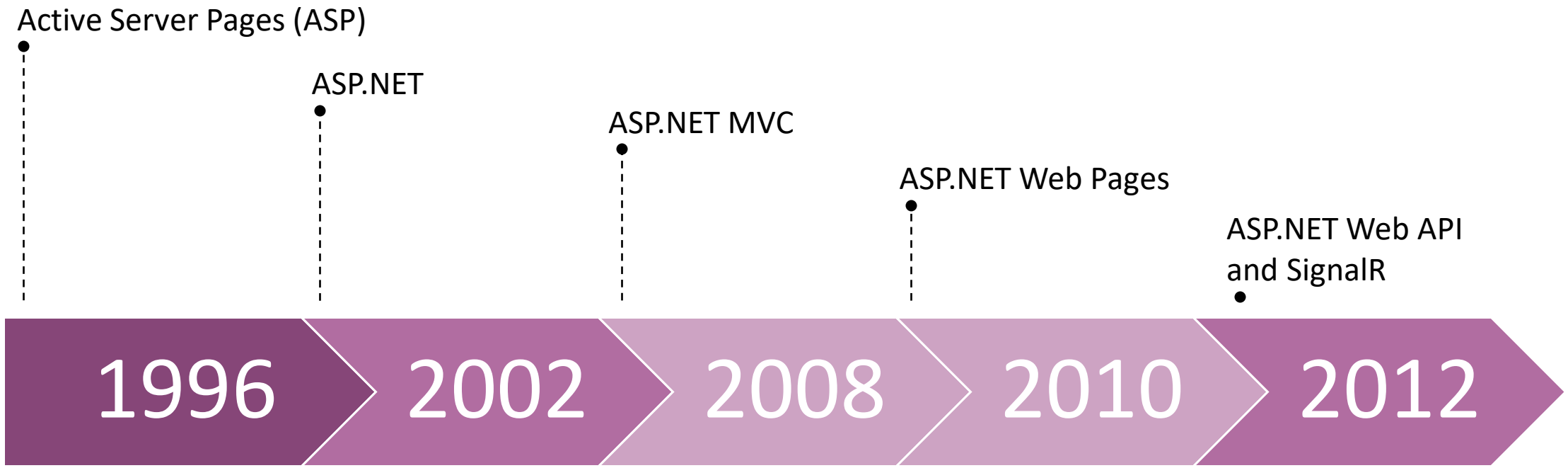
- Standards based
- Develop in browser
- Open tooling
- NPM / Bower
- Gulp / Webpack

ASP.NET Core 1.0 was previously called ASP.NET 5

It was renamed in January 2016

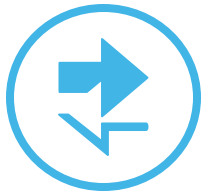
MVC and Web API are features of ASP.NET Core

History of ASP.NET



ASP.NET Core – Key values

Modular



Faster development cycle

Seamless transition to the cloud

Use favorite editors

Open Source

Cross Platform

Performance

Unified experience

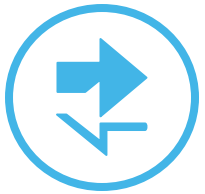
Productive

Unified experience

MVC and Web API unified (Web pages to come)

Dependency injection

Tracing and diagnostics



Faster development cycle

Features are shipped as packages via NuGet

Frameworks ships as part of the application

Side-by-side framework versioning

Service pack control

Fast

Performance

Faster startup times

Lower memory / higher density (> 90%)

ASP.NET Core exceeds 1.15 million requests pr. second

That is a 2300% more request than ASP.NET 4.6

Modular

IIS or self-hosted

New HTTP request pipeline utilizing Middleware

Opt-in to desired features

Flexible

Seamless transition to the cloud

Cloud ready configuration, session and cache

No code changes for cloud

Diagnostics, remote tracing and debugging

Use favorite editors

Visual Studio, Text editors and cloud editors

No editors (Command line)

Cross platform

Cross Platform

Windows

Linux

Mac

Open Source

Apache License version 2.0

Available on GitHub

Pull requests

ASP.NET Core – An introduction

Startup class

The request pipeline is configured by adding middleware components to an IApplicationBuilder instance.

- This is instance of IApplicationBuilder is provided via Dependency Injection

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseMvc();
}
```

Dependency Injection (DI)

DI is a first class citizen of .NET Core

Default Inversion of Control container (Can be replaced)

Declare services in startup.cs

This method gets called by the runtime. Use this method to add services to the container.

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

DI – Adding MVC

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

// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseMvc();
}
```

Views

Views now supports dependency injection → @inject

_ViewImports.cshtml used for registering namespaces in Views

```
@model HomeModel
@inject IDataService

@{
    ViewData["Title"] = "Index - MVC Example";
}

<h1>@Model.Headline</h1>

<p>
    @ExampleService.GetText()
</p>
```

Tag helpers

HTML helpers expressed as HTML attibutes

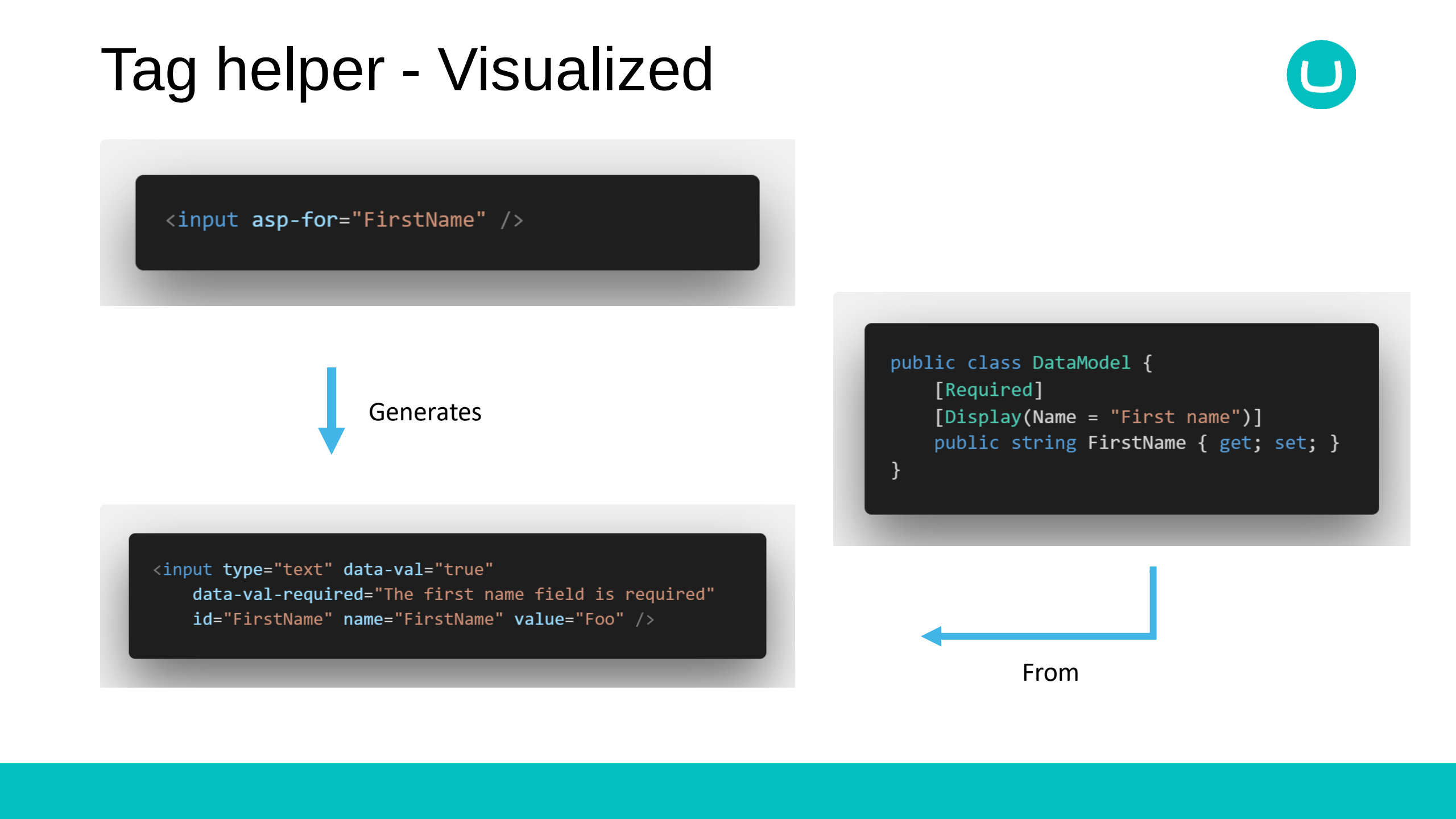
Designer friendly

Easier to customize with additional attributes

Works seamlesly with any HTML editor

```
<form asp-controller="Home">
    <label asp-for="Id"></label>
    <label asp-for="FirstName"></label>
    <input asp-for="FirstName" />
    <label asp-for="LastName"></label>
    <input asp-for="LastName" />
    <input type="submit" value="Go!" />
</form>
```

Tag helper - Visualized



Custom tag helper

```
public class CustomTagHelper : TagHelper
{
    public string Title { get; set; }
    public string Body { get; set; }

    public override void Process(TagHelperContext c, TagHelperOutput output)
    {
        output.TagName = "p";
        output.Attributes.Add("class", "example");
        output.PostContent.SetContent("Hello from my custom tag helper");
        output.PostContent.Append($"Title is: {Title} and body is: {Body}");
    }
}
```

ASP.NET Core demo

Visual Studio…..

MVC vs. Web API

MVC

Designed create websites

Returns both views and data

Only returns data as JSON

Web API

Web API is intented to create full blown HTTP services

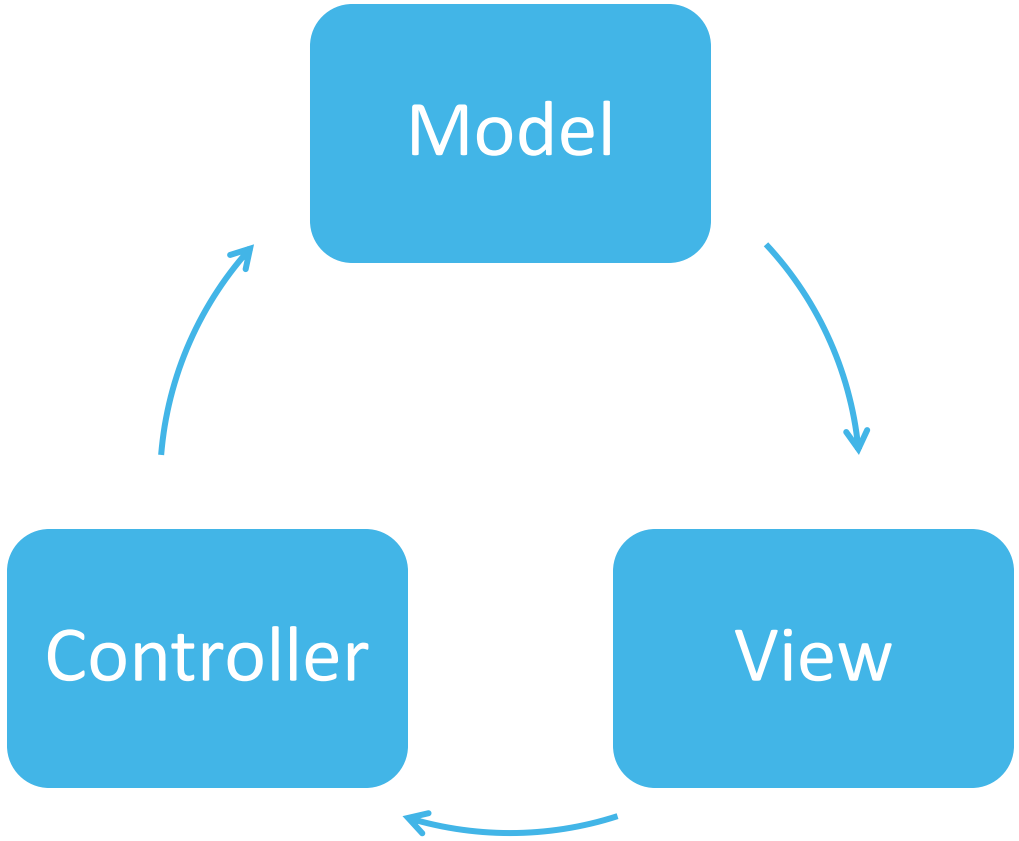
Supports content negotiation

Returns data as JSON, XML, etc.

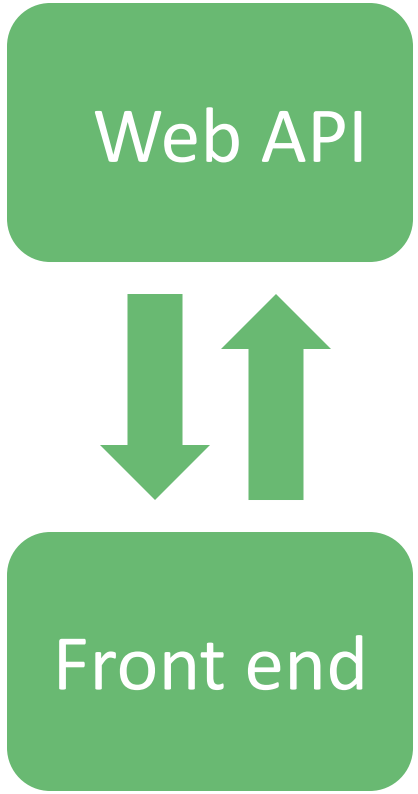
Can be used for more than web

Web API and MVC can be combined!

MVC



Web API



Exercise Time!

Time: 60 minutes

Headline: Make a small website

Purpose: Learn how to make a website using ASP.NET MVC

How: Create a project using Visual Studio or dotnet CLI

Requirements:

- Front page
  - With links to About me and Gallery pages
- About me page
- Gallery page
- All data for the pages should come from Controllers

Entity Framework Core

Entity Framework Core 2.0

Previously named Entity Framework 7

Rewritten from scratch

Common classes, patterns and workflows remain intact

Code first workflow only – NO EDMX Designer mode! (Which is good…)

Smaller footprint

Cross platform

Entity Framework Core 2.0 (cont.)

EF now targets .NET Standard 2.0

It has improved LINQ translation

Like query operator introduced

Entity Framework Core 2.0 – Dark side

EF Core does not support complex types

- Instead it has “owned” or “child” types

Lazy loading is being delayed until EF Core 2.1

No full support for Stored Procedures

Questions?

Midway evaluation

Now over to you!

I’m leaving the room

Choose a minute taker and one as a moderator

Evaluate the f*** out of me! ;)

½ - 1 hour

Ideas…

What did I do that was good?

What could be better?

What should I absolutely stop with?

Are my teaching style okay?

Etc.???

Questions?