

ASP.NET Core 1.0

SOMEONE MOVED YOUR CHEESE

J. Tower

About Me

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

Microsoft MVP in ASP.NET

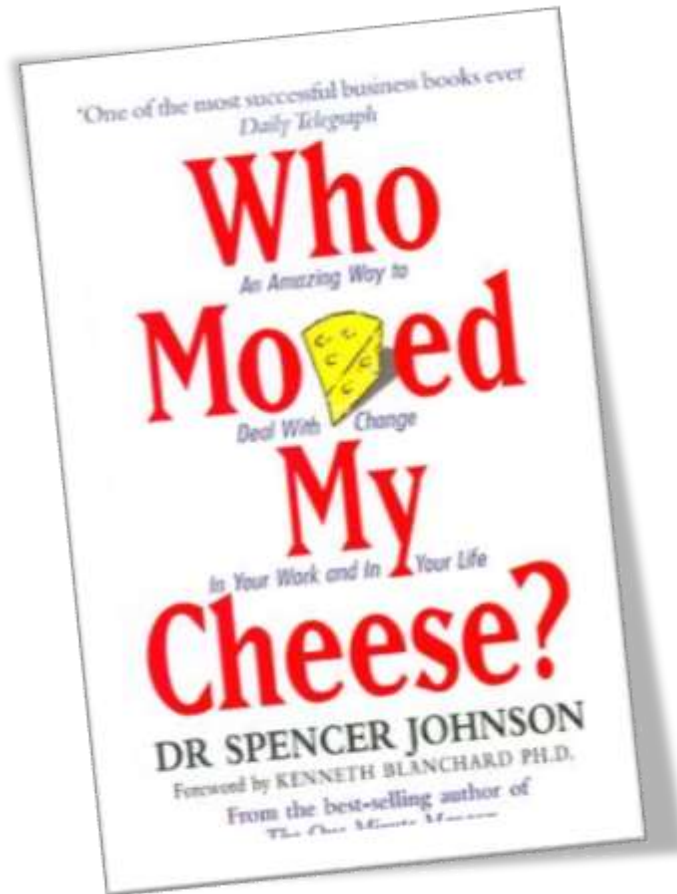
✉ jtower@trailheadtechnology.com

🌐 trailheadtechnology.com/blog

🐦 [jtowermi](https://twitter.com/jtowermi)



A Little About The Session Title



Who Moved My Cheese, by Dr Spencer Johnson

Change Happens

They Keep Moving The Cheese

Anticipate Change

Get Ready For The Cheese To Move

Monitor Change

Smell The Cheese Often So You Know When It Is Getting Old

Adapt To Change Quickly

The Quicker You Let Go Of Old Cheese, The Sooner You Can Enjoy New Cheese

Change

Move With The Cheese

Enjoy Change!

Savor The Adventure And Enjoy The Taste Of New Cheese!

Be Ready To Change Quickly And Enjoy It Again

They Keep Moving The Cheese.x

Overview

.NET Version History

.NET Core

CLI and X-Platform

Project Structure Changes

Pipeline

Tag Helpers

DI Framework

Client-Side Development
Improvements

Let's Get Started, But First...

If You Give \$75, So Will I!

bit.ly/aspnet-ccc

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 13,641 projects in 22 countries, benefiting over 4.6 million people." - Wikipedia

*"97.1/100"
- CharityNavigator.org*



charity: water

ASP.NET Version History

History of ASP.NET

Version	.NET / IDE Version	Key Features
1.0	1.0 and Visual Studio .NET	OO, event-driven web apps, viewstate, DLL class libraries
1.1	1.1 and Visual Studio .NET 2003	ODBC
2.0	2.0 and Visual Studio 2005	New controls, Themes, Skins, Master pages, Membership Services, Localization
3.5	3.5 and Visual Studio 2008	ASP.NET AJAX, LINQ, MVC 1
4.0	4.0 and Visual Studio 2010	Routing
4.5	4.5 and Visual Studio 2012	Unobtrusive Validation, Async, HTML5, WebSocket
4.5.1	4.5.1 and Visual Studio 2013	One ASP.NET, Scaffolding, Identity

What's Next

V4.6?

v5?

Something
else?

A hiker wearing a blue jacket, dark pants, a wide-brimmed hat, and a large backpack stands at a fork in a dirt road. The road splits into two paths: one to the left labeled 'v4.6' and one to the right labeled 'v1'. The hiker is positioned in the center, facing away from the camera. The background shows a hilly, wooded landscape with green grass and scattered trees under bright sunlight.

v4.6

v1

A Fork in the Road

I Thought It Was ASP.NET 5

Well, about that...

ASP.NET 5, We Hardly Knew Ye





ASP.NET 5 now becomes **ASP.NET Core .10**

EF 7 becomes **EF Core 1.0**

Core 1.0 – because it's new, different, and built for .NET Core

ASP.NET – because it's still so familiar

ASP.NET 4.6 and ASP.NET Core 1.0

ASP.NET 4.6	ASP.NET Core 1.0
.NET Framework 4.6 	.NET Core 1.0   
.NET framework libraries	.NET core libraries
Compilers and runtime components (.NET Compiler Platform: Roslyn, C#, VB, F# Languages, RyuJIT, SIMD)	

Source: <http://www.hanselman.com/blog/ASPNET5IsDeadIntroducingASPNETCore10AndNETCore10.aspx>

What is .NET Core?

.NET Core

Completely Rewritten

Open Source

Cross-platform: Linux, Mac, and Windows

Includes

- Run-time and JITer

- Native Compilation (!!!)

- Base Class Libraries (BCL)

- .NET Command Line Interface (CLI)

Comparing Frameworks

.NET Framework

Common Language Run-time (CLR)
& Base Class Library (BCL)

Run by the OS

Mature and fully featured framework

Ships with Windows, but runs only on Windows

Monolithic, large API surface area

Slower release cycle

Available for reference, but not strictly open source

.NET Core

CoreCLR
& CoreFX (basically, the Core BCL)

Run by the .NET Execution Environment (DNX)

New, rewritten and limited

Windows, Mac, and Linux

Fully Modular

Rapid updates

Fully open source project

ASP.NET 4.6

Based on .NET 4.6

WebForms

System.Web

Model-binding with async

.NET compiler platform

HTTP/2 support

Identity Updates

JSON, JavaScript, HTML editor improvement

JSX (React.JS) support

ASP.NET Core 1.0

Can run on the full .NET Framework
or on .NET Core

New light-weight request pipeline

Run on IIS, or self-hosted in your own process

Runs cross-platform on Windows, Mac, and
Linux

Everything is a package delivered with NuGet,
even the runtime itself

Comes with MVC 6 - a unified Web framework
for Web UI and Web APIs

Environment-based configuration for a
seamless transition to the cloud

Dependency injection out-of-the-box

New Visual Studio project system and high
productivity tooling experience

All open source on GitHub through the .NET
Foundation

Why ASP.NET Core 1.0?

New light-weight and modular HTTP request pipeline

Ability to host on IIS or self-host in your own process

Built on .NET Core, which supports true side-by-side app versioning

Ships entirely as NuGet packages

Integrated support for creating and using NuGet packages

Single aligned web stack for Web UI and Web APIs

Cloud-ready environment-based configuration

Built-in support for dependency injection

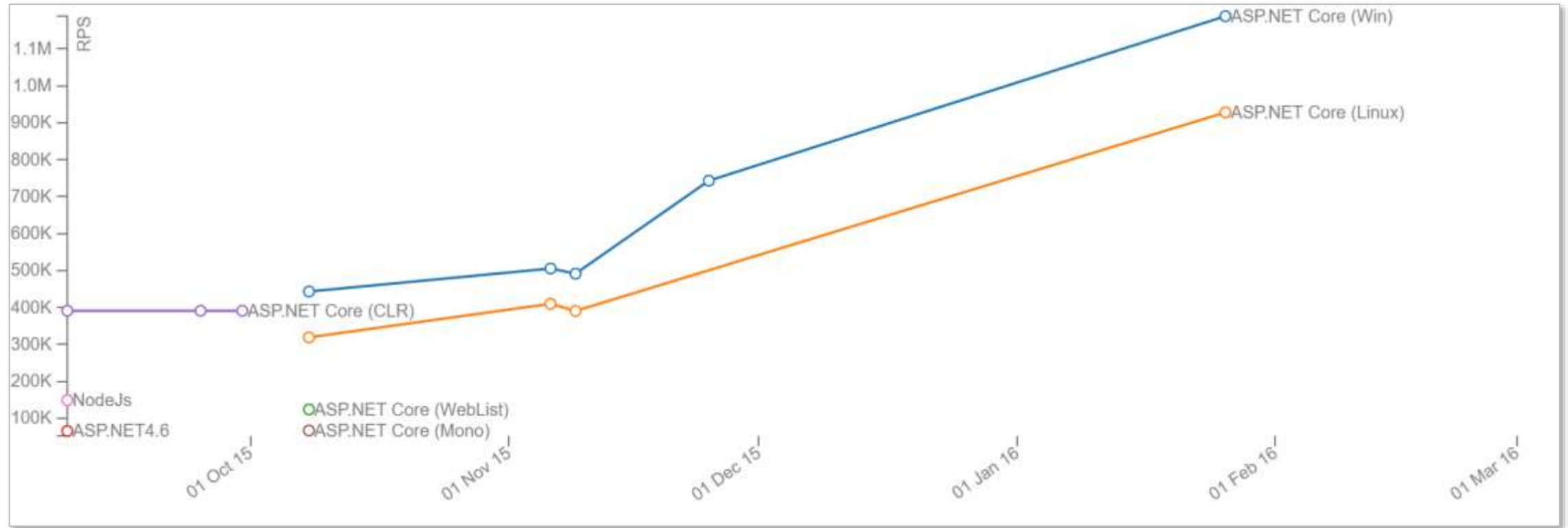
New tooling that simplifies modern web development

Open source and community focused

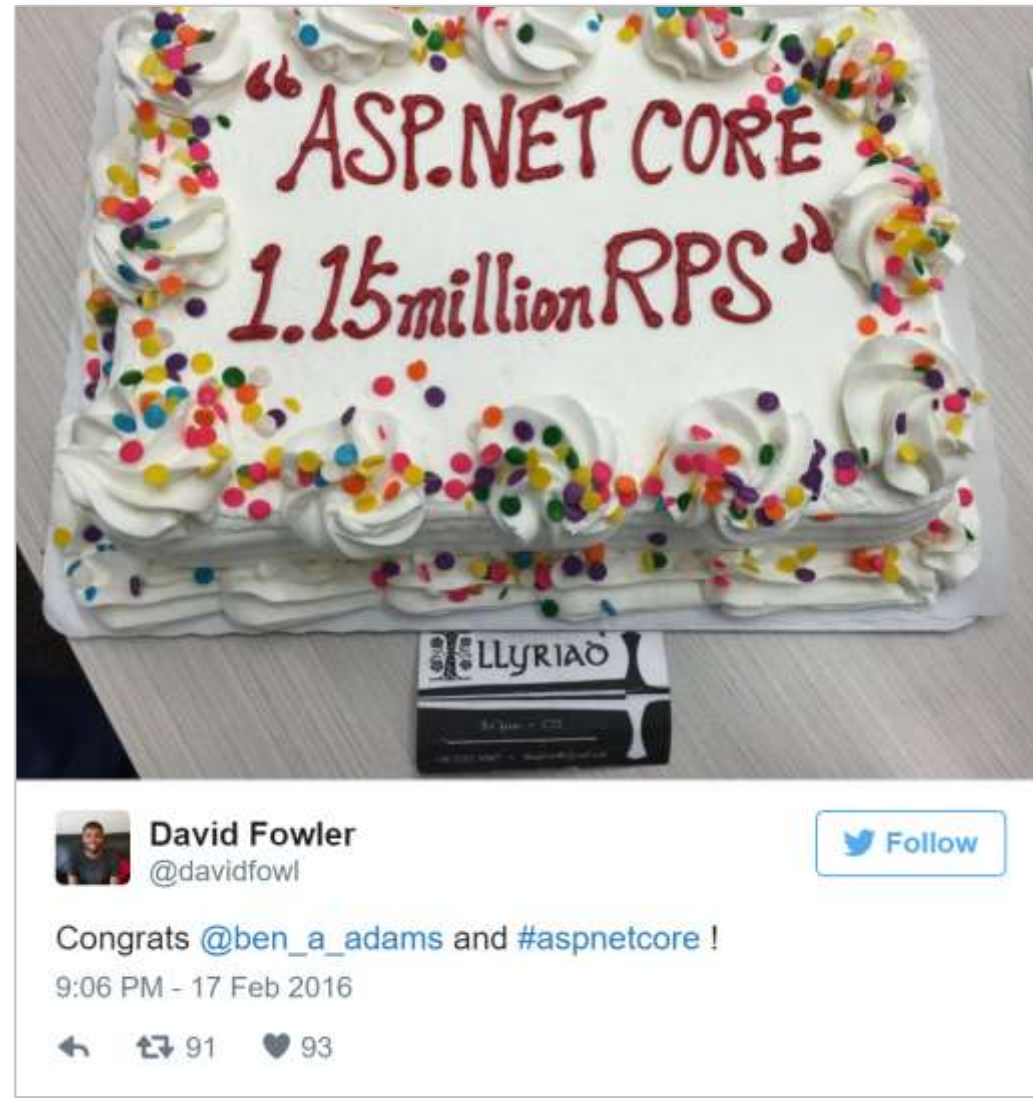
Build and run cross-platform ASP.NET apps on Windows, Mac and Linux

2K object graph per request (vs 30K)

Perf!



Perf!



ASP.NET Release Schedule

Milestone	Release
Beta6	27 Jul 2015
Beta7	2 Sep 2015
Beta8	15 Oct 2015
RC1	18 Nov 2015
RC2	16 May 2016
1.0.0	late-June 2016
1.?.? *	Q3 2016

*Support for Visual Basic, SignalR 3, or Web Pages 4

.NET CLI and Cross-Platform

.NET CLI

```
dotnet run  
dotnet compile  
dotnet pack  
dotnet restore  
dotnet publish
```


Cross-Platform ASP.NET

get.asp.net

ASP.NET Core 1.0 Project Structure

Framework Target

WebApplication1* [icon] [X]

Application* [arrow]
Build
Debug

Configuration: N/A Platform: N/A

Name: WebApplication1

Default namespace: WebApplication1

☒ Use specific DNX version:

Version	Platform	Architecture
1.0.0-beta5-11584	.NET Framework	x86
1.0.0-beta5-11584	.NET Framework	
1.0.0-beta5-11565	.NET Core	
1.0.0-beta4-11296		
1.0.0-beta4		

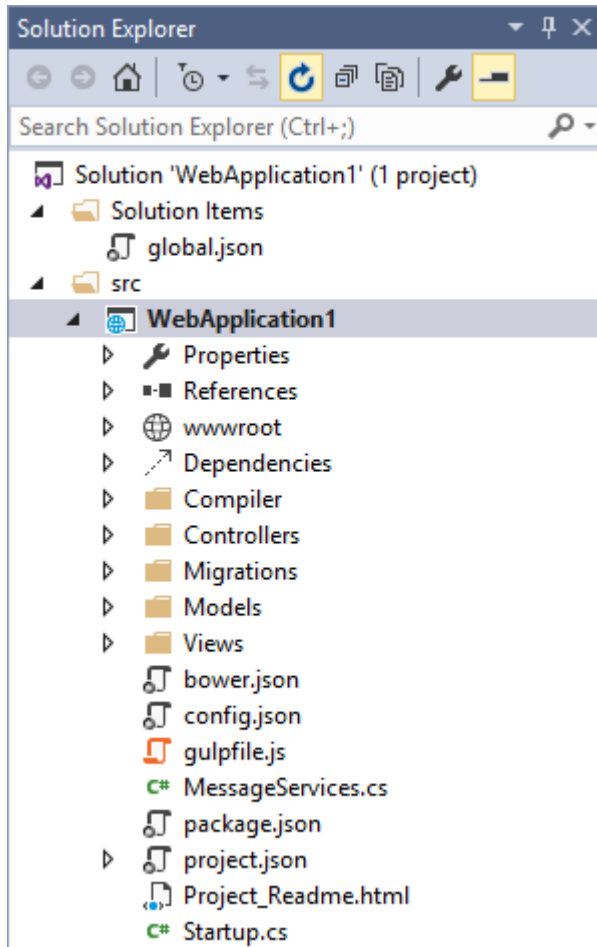
Web root:

project.json File (continued)

```
{  
  "webroot": "wwwroot",  
  "version": "1.0.0-*",  
  "dependencies": ...,  
  "commands": ...,  
  "frameworks": ...,  
  "exclude": ...,  
  "bundleExclude": ...,  
  "scripts": ...  
}
```

webroot:	the folder that is the root of the web site
version:	current project version
authors:	project author(s)
description:	project description
dependencies:	the NuGet section
commands:	custom commands for command-line
frameworks:	DNX versions supported
exclude:	files/folders to exclude from builds
bundleExclude:	files/folders to exclude from bundles
scripts:	build automation events and scripts

project.json File (continued)

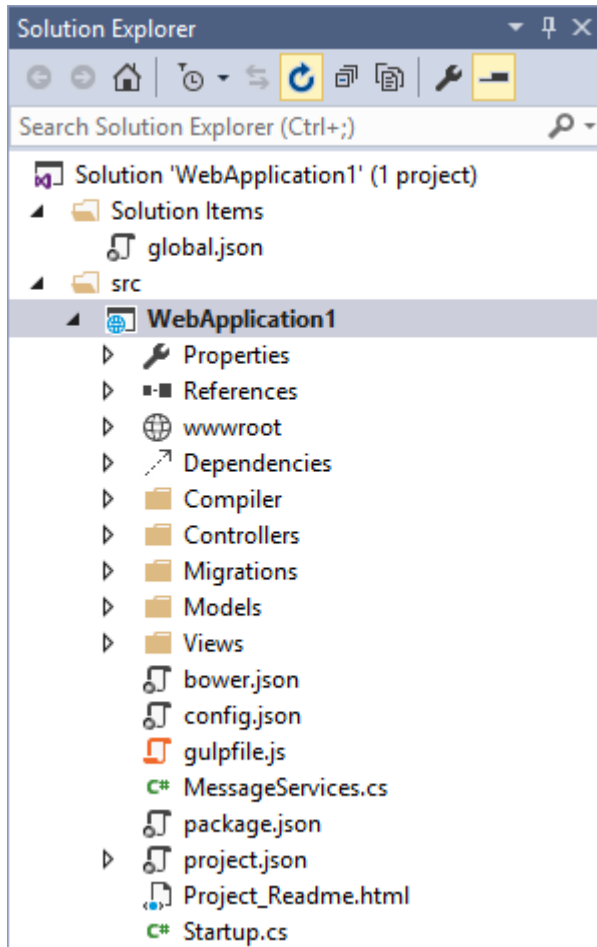


Server-side dependencies
(AKA NuGet packages)

packages.config → project.json
XML → JSON

Managed manually or via NuGet UI

global.json File



Configure the solution as a whole:

'projects' Section

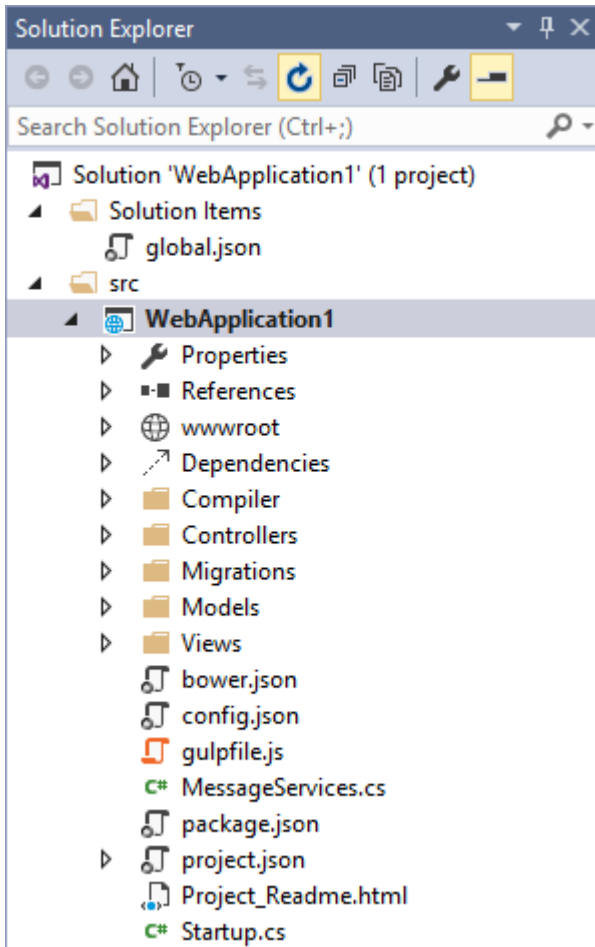
Contains web project locations ("src")

'sdk' Section

Contains target DNX

NOTE: Visual Studio still produces a .sln file

wwwroot Folder

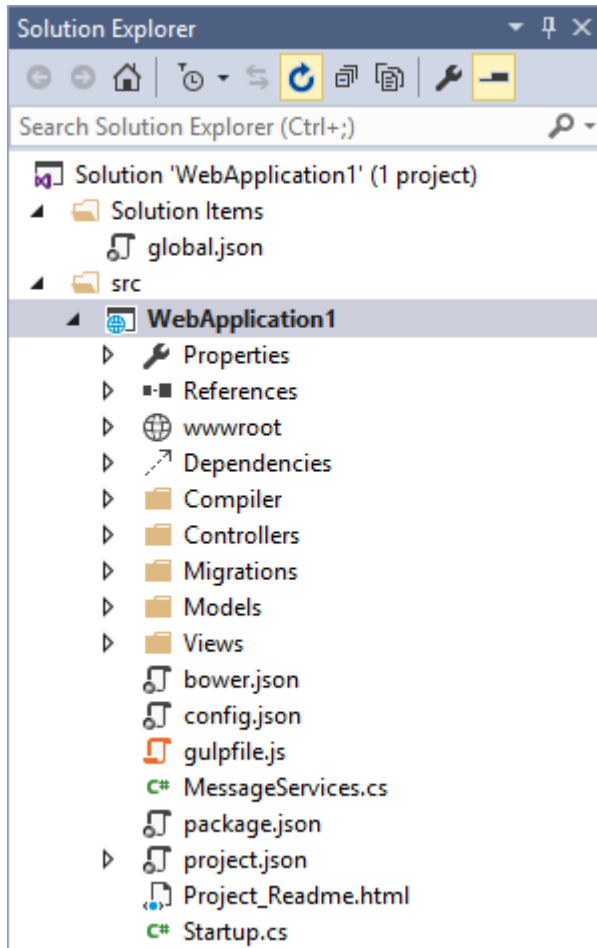


Root folder from which files are served

Protects sensitive files

Simplifies bundling/minification

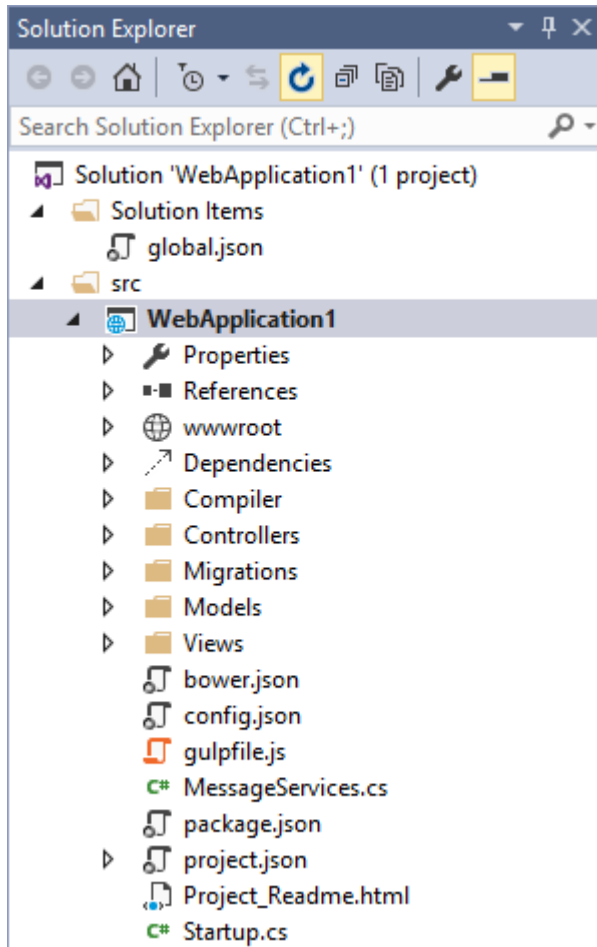
Client-Side Dependency Management



Support for
NPM
Bower
(NuGet)

More in “Client-Side” Section

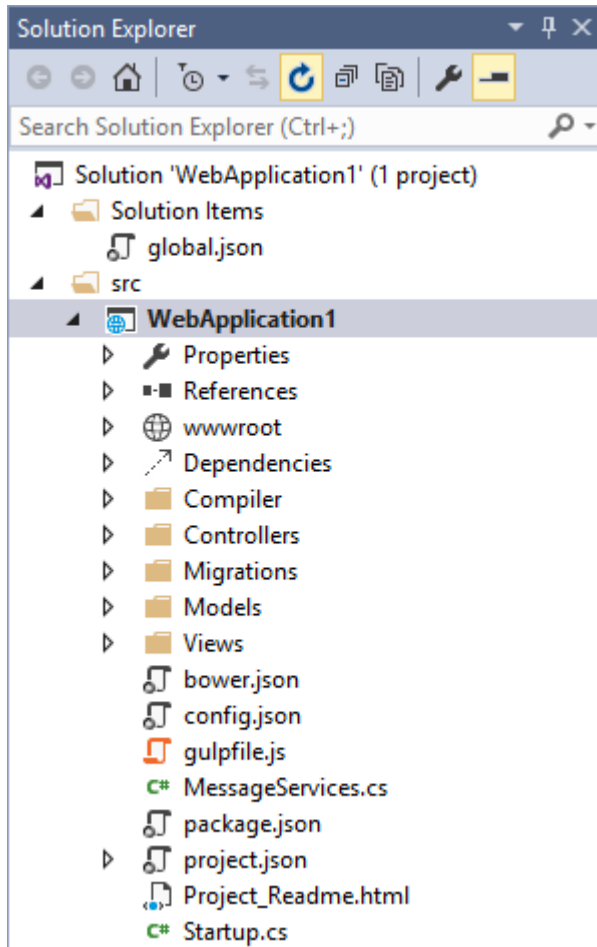
Client-Side Task Runners



Support for
Grunt – gruntfile.js
Gulp – gulpfile.js

More in “Client Side” section

Configuring the Application



Web.Config -> *.json

New configuration model for name-value pairs

Not based on System.Configuration or web.config

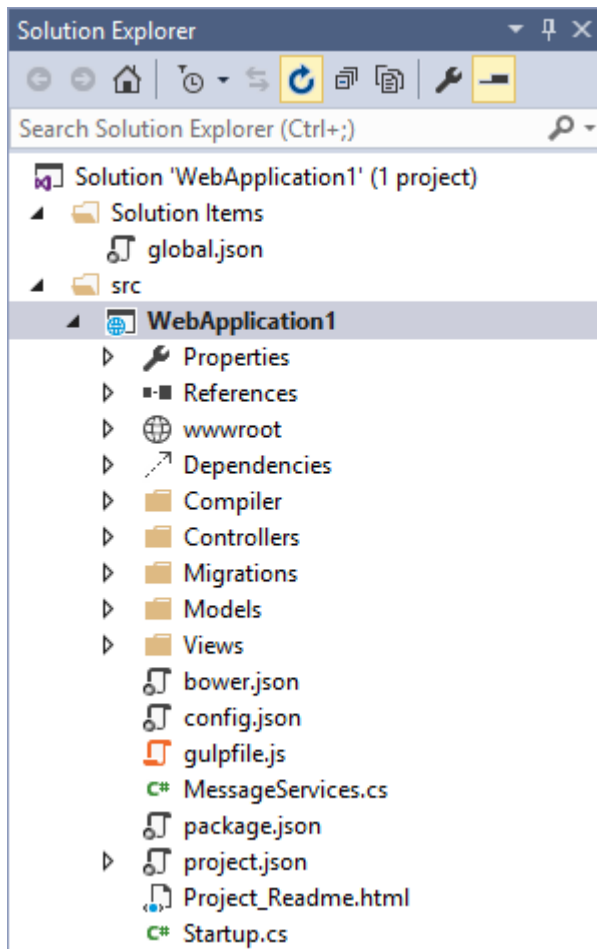
Providers support XML, JSON, INI, and environment variables

Custom configuration providers

Environments (Dev/Prod) are a first-class notion in ASP.NET

Nuget: packages.config → packages.json

Startup



ASP.NET Core 1.0 applications are defined using a public Startup class:

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

    public void Configure(IApplicationBuilder app)
    {
    }
}
```

ConfigureServices: defines services used by your application

Configure: defines middleware making up request pipeline

ASP.NET Core Pipeline

Hosting

Microsoft.AspNet.Server.IIS

Microsoft.AspNet.Server.WebListener (WebListener)

Microsoft.AspNet.Server.Kestrel (Kestrel)

Self Hosting

Run web site as a service. No need for IIS/IIS Express!

1. Add NuGet Packages

Microsoft.AspNet.Hosting

Microsoft.AspNet.Server.WebListener

2. Add command

```
"commands": {  
    "web": "Microsoft.AspNet.Hosting  
           --server Microsoft.AspNet.Server.WebListener  
           --server.urls http://localhost:5000"  
},
```

Goodbye, HttpModules & HttpHandlers

HTTPModules replace by Middleware

Pass-through components that form a pipeline between a server and application to inspect, route, or modify request and response messages for a specific purpose.

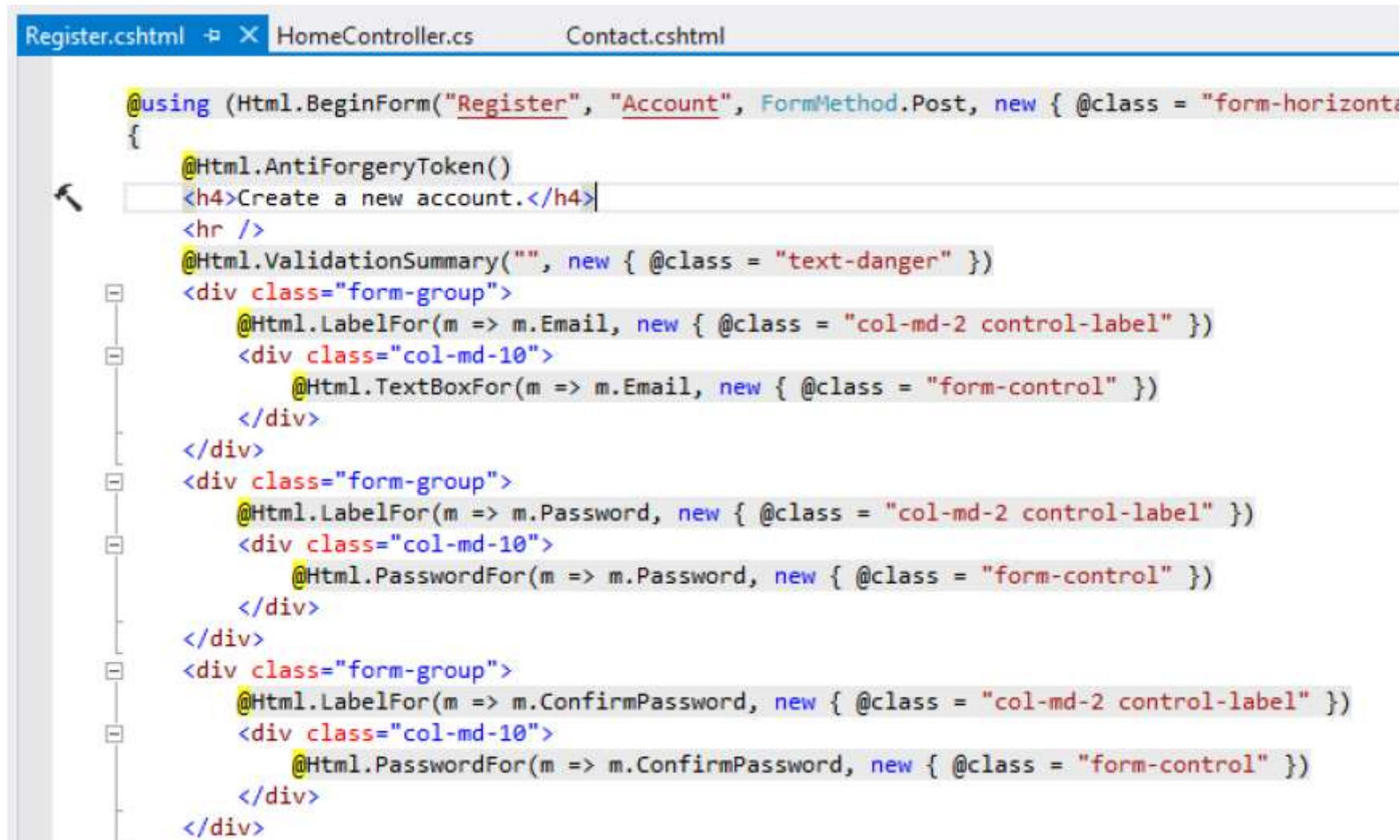
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
}
```

```
public void Configure(IApplicationBuilder app)
{
    app.UseMvc( /*config here*/ );
}
```

Tag Helpers

New in MVC 6

Html Helpers



```
Register.cshtml  X  HomeController.cs  Contact.cshtml

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizontal" })
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
}
```

Built-In Tag Helpers

```
<environment names="Development">
    //raw resources
</environment>
<environment names="Staging,Production">
    //minified, bundled resources
</environment>

<cache expires-after="@TimeSpan.FromMinutes(5)" vary-by-user="true">
    @Html.Partial("_PartialView.cshtml")
</cache>



<a asp-action="Index" asp-controller="Home">Back to Home</a>
```

Custom Tag Helpers

```
[HtmlTargetElement("p", Attributes = "my-tag")]
[HtmlTargetElement("my-tag")]
public class MyTagHelper : TagHelper
{
    public async override Task ProcessAsync(TagHelperContext context,
        TagHelperOutput output)
    {
        var childContent = await context.GetChildContentAsync();
        var markdownContent = childContent.GetContent();

        output.Content.SetContentEncoded("<h2>Hello, world!</h2>");
    }
}
```

<my-tag></my-tag>

<p my-tag></p>

Tag Helper Demo

DEMO: Angular2 Pre-Rendering

Dependency Injection Framework

Dependency Injection

Inversion of Control (IoC)

Built-in framework

Services

- reusable components

- injected in application where needed

- Loose coupling

Services

```
public class Startup
{
    ...

    public void Configure(IApplicationBuilder app)
    {
        app.UseServices(services =>
        {
            services.AddTransient<IRepository, DataRepository>()
        });
    }
}
```

Services

```
public interface IEmailSender
{
    Task SendEmailAsync(string email, string subject, string message);
}
```


Services

```
public class HomeController : Controller
{
    IEmailSender _emailSender;

    public HomeController(IEmailSender emailSender)
    {
        _emailSender = emailSender;
    }

    [HttpPost]
    public async Task<ActionResult> Contact(ContactViewModel model)
    {
        await _emailSender.SendEmailAsync(model.Email, model.Subject, model.Message);
        return View();
    }
}
```

Dependency Life-Times

Instance: a specific instance is given all the time; you are responsible for its initial creation

Singleton: single instance throughout the whole application life-time

Scoped: single instance within the scoped container

Transient: a new instance every time the service is requested

Client-Side Development

Client-side Tools Support

Gulp & Grunt – Task runners

NPM & Bower – Client-side package manager

DEMO

- Use NPM to add gulp-less

- Compile less->css with gulp and VS integration

- Adding client-side component with Bower

Interested in More?

www.asp.net/vnext

docs.asp.net

get.asp.net

channel9.msdn.com

trailheadtechnology.com/blog

jtower.com/blog

If You Give \$75, So Will I!

bit.ly/aspnet-ccc

"charity:water is a non-profit organization that provides clean and safe drinking water to people in developing nations. The organization was founded in 2006 and has helped fund 13,641 projects in 22 countries, benefiting over 4.6 million people." - Wikipedia

*"97.1/100"
- CharityNavigator.org*



charity: water

Thank You! Questions?

Jonathan "J." Tower

Principal Consultant & Partner

Trailhead Technology Partners

Microsoft MVP in ASP.NET

✉ jtower@trailheadtechnology.com

🌐 trailheadtechnology.com/blog

🐦 [jtowermi](https://twitter.com/jtowermi)

