ASP.NET 5

SOMEONE MOVED YOUR CHEESE

J. Tower

About Me

Jonathan “J.” Tower

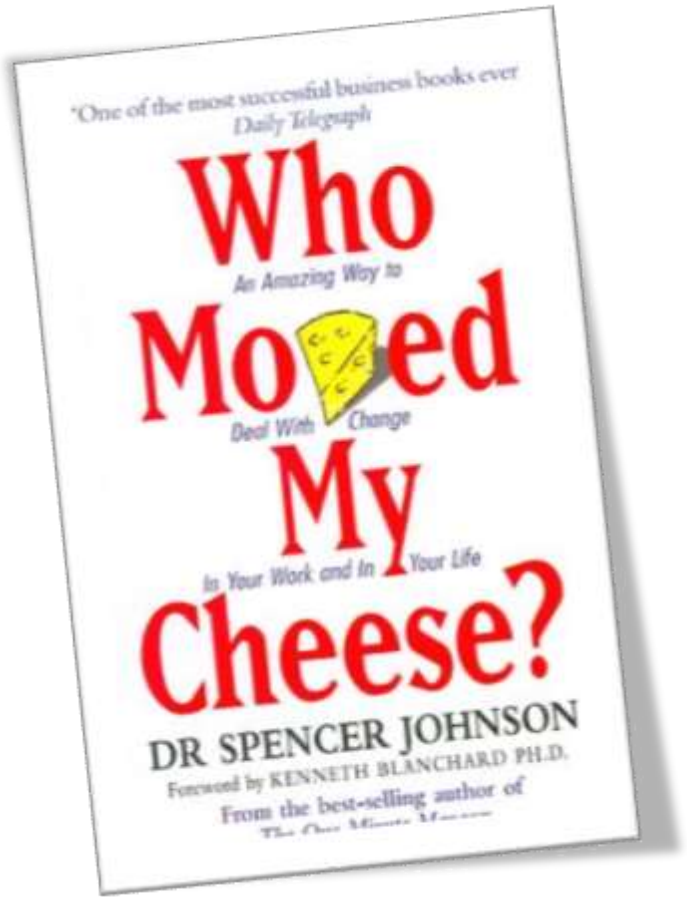
Principal Consultant, JTower.com

jtower@jtower.com

jtower.com/blog

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

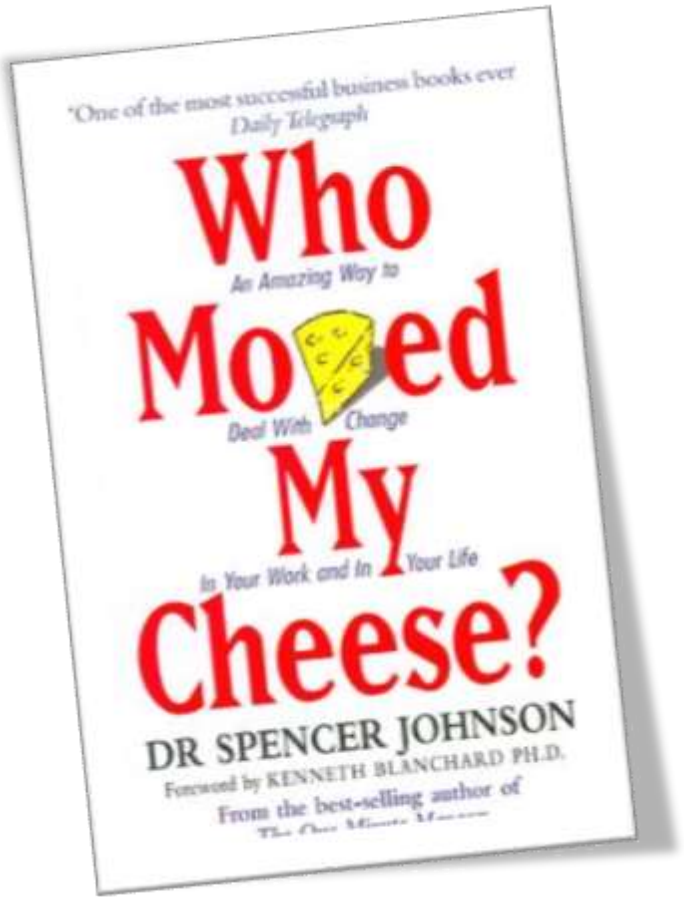
Adapt To Change Quickly
The Quicker You Let Go Of Old Cheese, The Sooner You Can Enjoy New Cheese

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

A Little About The Session Title



Who Moved My Cheese, by Dr Spencer Johnson

Change

Enjoy Change!

Be Ready To Change Quickly And Enjoy It Again

Overview

Versions & History of ASP.NET

ASP.NET Command Line

Project Structure

Pipeline

Tag Helpers

Dependency Injection Framework

Client-side Development

Let's Get Started, But First...

If You Give $100, So Will I!

http://bit.ly/detroit-gives

"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

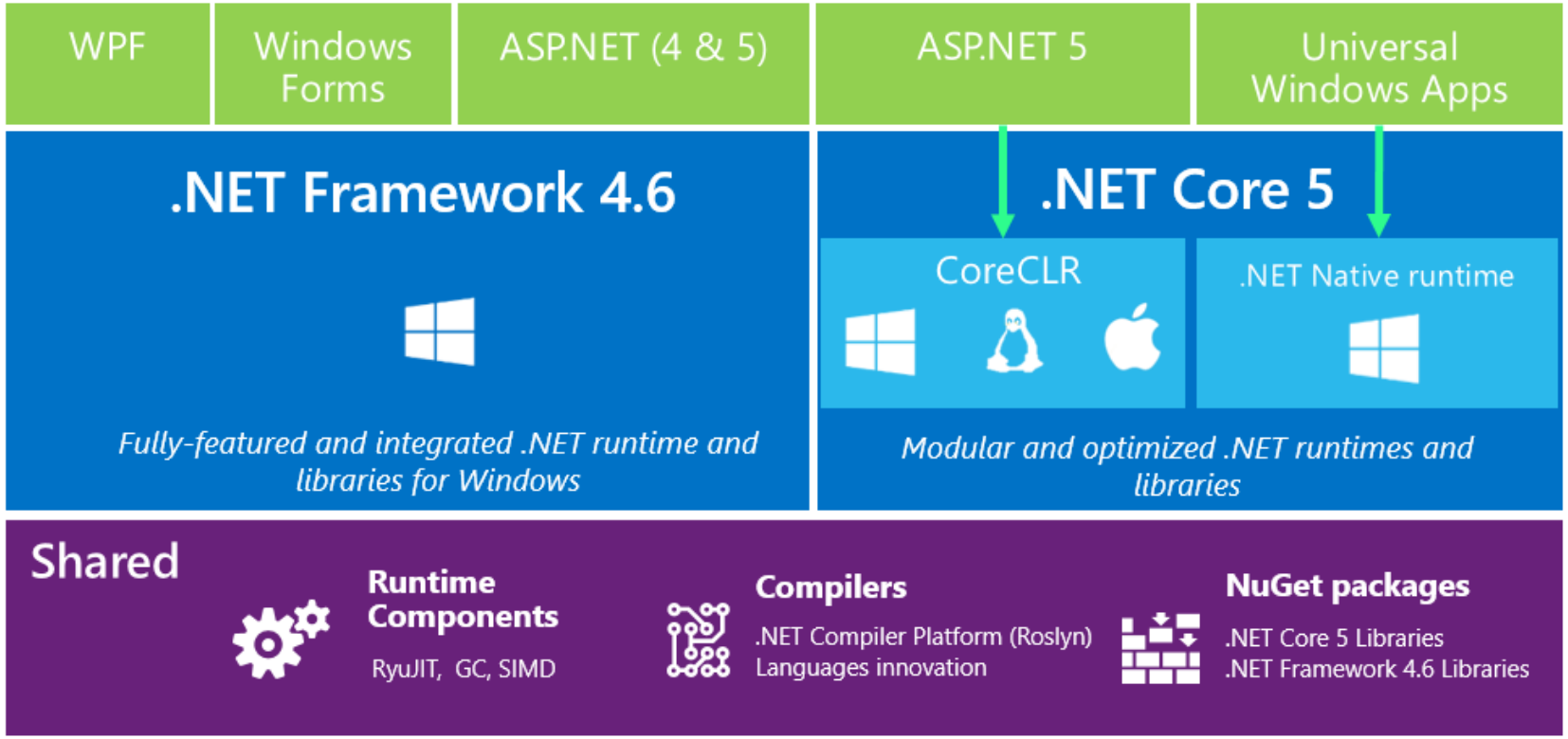
charity: water

ASP.NET
Versions & History

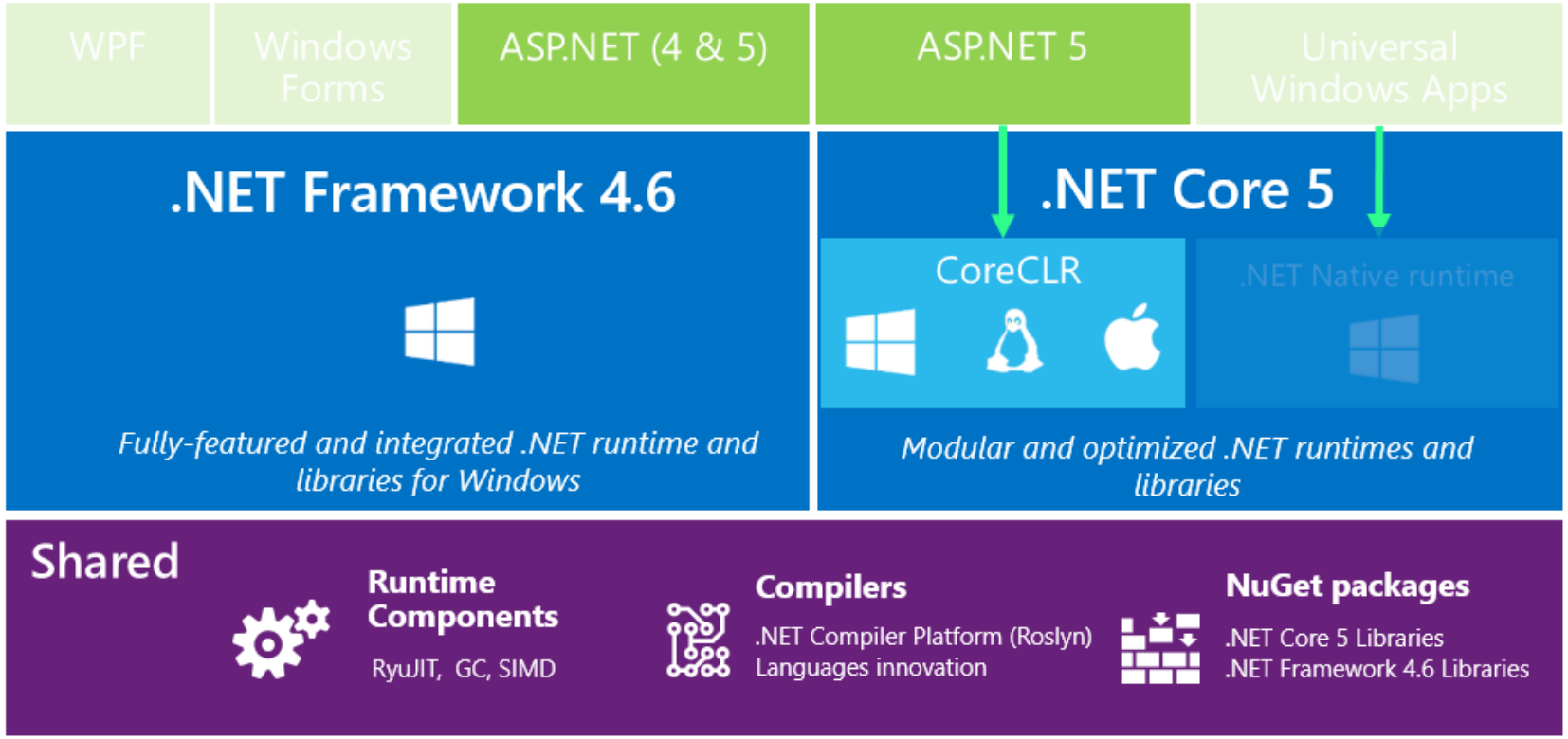
History of ASP.NET

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

.NET 4.6 and .NET Core 5



.NET 4.6 and .NET Core 5



CLR & BCL

Common Language Run-time & Base Class Library

AKA .NET Framework

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 and CoreFX

.NET Core consists of CoreFX & CoreCLR

CoreCLR
- Small, optimized runtime
- Can be targeted by ASP.NET 5 applications. (default)
- Modular runtime and library implementation

CoreFX
- Subset of the .NET Framework BCL

.NET Core is all open-source

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 5

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

ASP.NET Release Schedule

Milestone	Release week
Beta6	27 Jul 2015
Beta7	2 Sep 2015
Beta8	15 Oct 2015
RC1	Nov 2015
RC2	?
1.0.0	Q1* 2016

Why ASP.NET 5?

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)

ASP.NET 5
Command Line

DNVM

.NET Version Manager

Used to install and manage multiple DNX versions

Lets you:

- List the different DNX versions installed
- Install new ones
- Switch the active DNX

DNX

.NET Execution Environment

A software development kit (SDK) and runtime environment

Everything you need to build and run .NET applications

Runs on Windows, Mac and Linux

Provides a host process, CLR hosting logic and managed entry point discovery

Built for running cross-platform ASP.NET Web applications

Can run other types of .NET applications, such as cross-platform console apps.

*Old Names

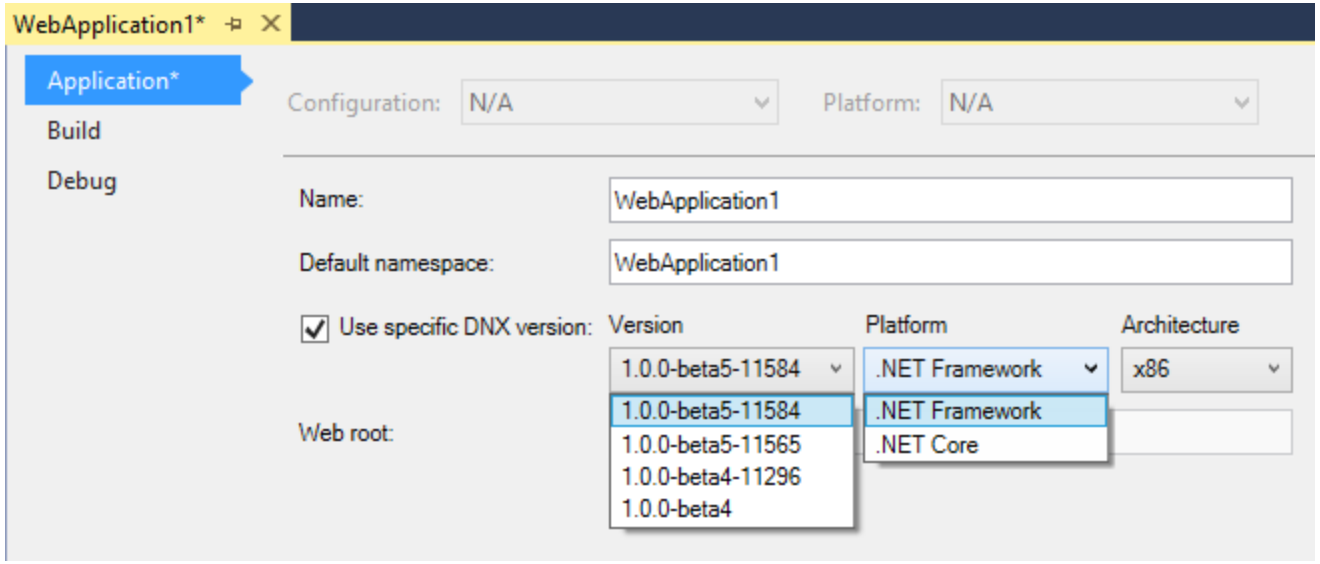
Current name	New name	Comments
k.cmd / klr.exe	dnx	The main runner entry point to boot an app / site. This was previously split into two files, but will be unified into one. Running “dnx.exe --help" (or the equivalent *nix/MacOS command) would show a help screen such as: C:\> dnx.exe --help Microsoft .NET Execution environment vX.Y.Z Usage: dnx [options] [command] [arguments] Options: --help|-h Show help Commands: run Run the main entry point of the application cmd1 Do cmd1
KRE / XRE	DNX	The .NET Execution environment – these are NuGet packages that are per arch/OS/CLR that contain the CLR (for Core CLR), CLR bootstrappers, and required app execution components.
kvm.ps1/.sh	dnvm.ps1/.sh	.NET (DNX) Version Manager. Manages which DNXs are installed, which DNX is currently in use, etc.
kpm	nuget	The package manager for NuGet packages in the app. Being merged with NuGet.

*New Name

dotnet

ASP.NET 5
Project Structure

Framework Target

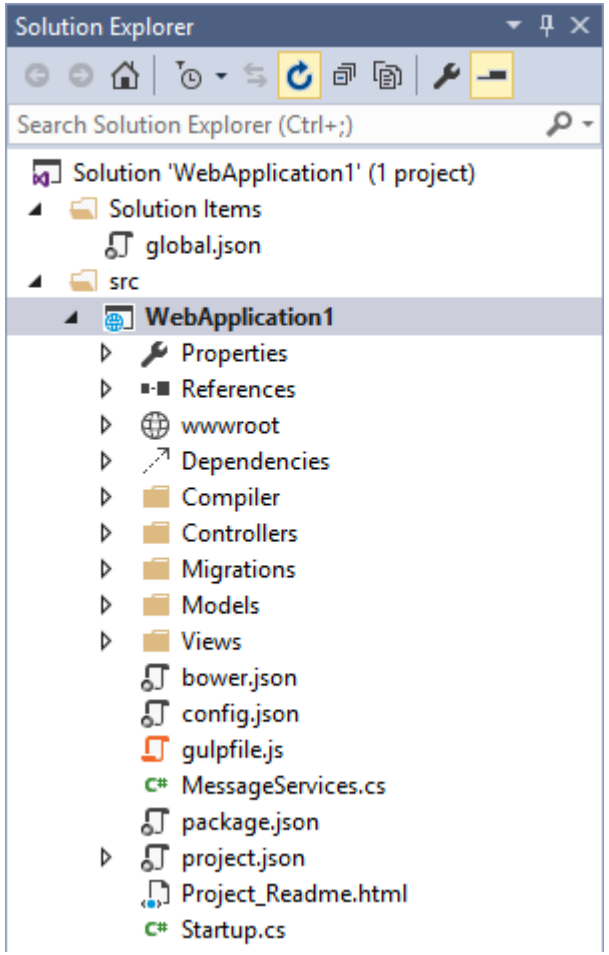


project.json File (continued)

```
{
    "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

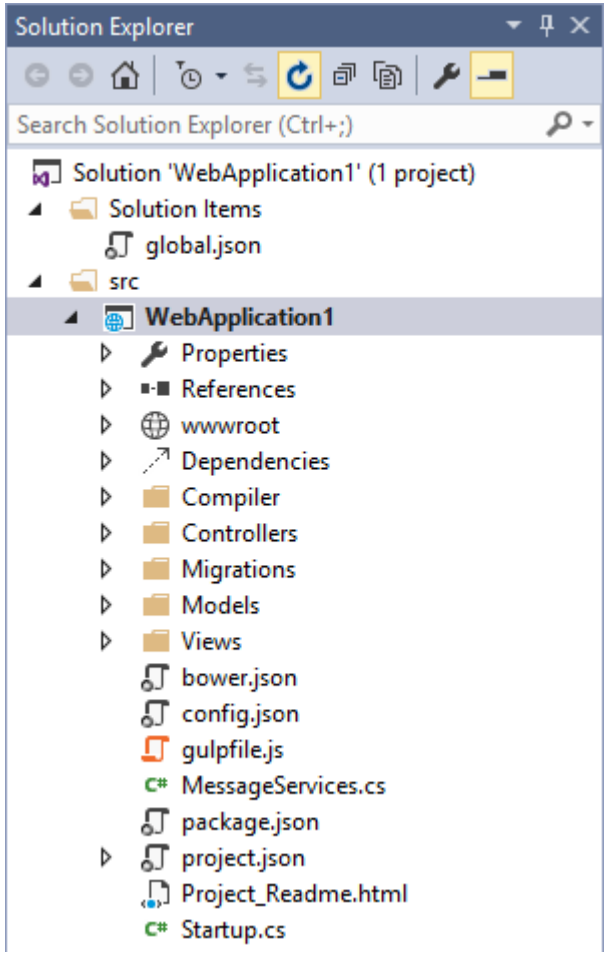
(AKA NuGet packages)

packages.config → project.json

XML → JSON

Managed manually or via NuGet UI

global.json File

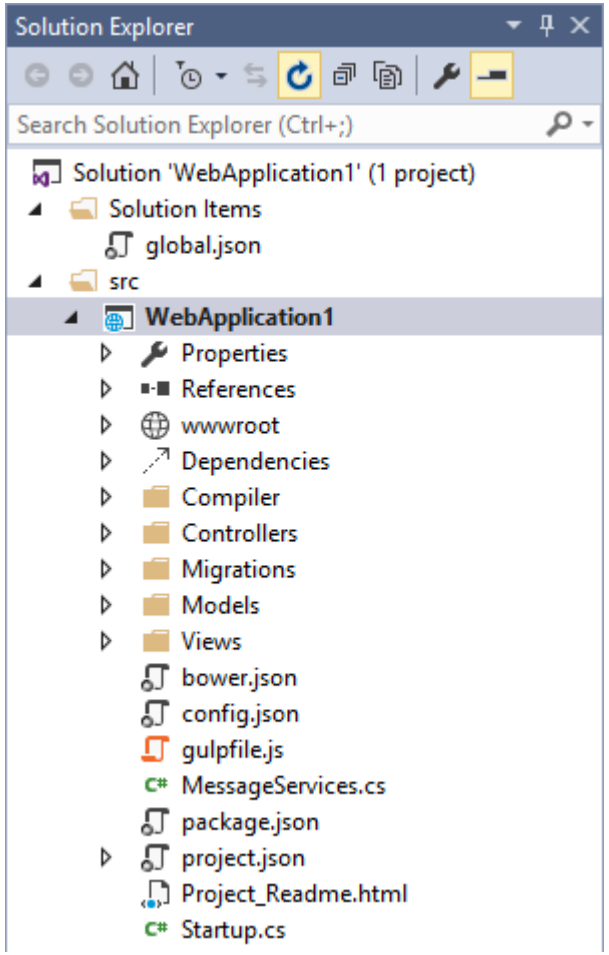


Configure the solution as a whole:

- 'projects' Section
  - *Contains web project locations ("src")*
- 'sdk' Section
  - *Contains target DNX*

NOTE: Visual Studio still produces a .sln file

wwwroot Folder

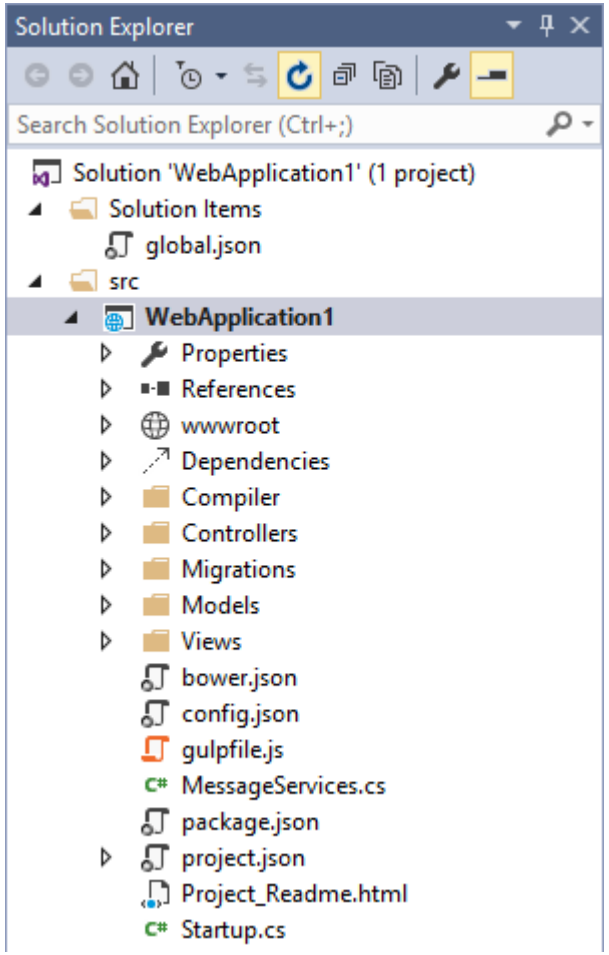


Root folder from which files are served

Protects sensitive files

Simplifies bundling/minification

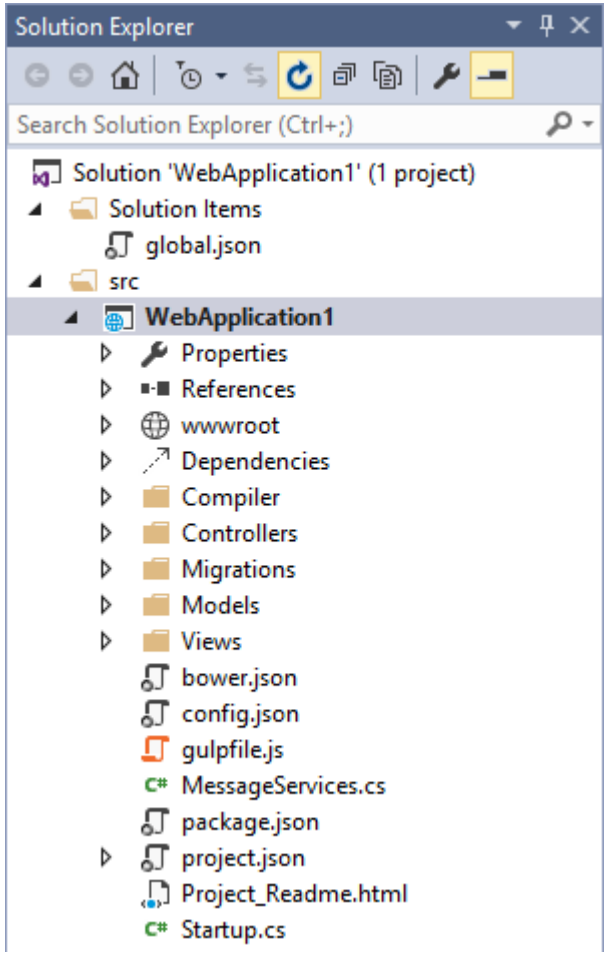
Client-Side Dependency Management



Support for
- NPM
- Bower
- (NuGet)

More in “Client-Side” Section

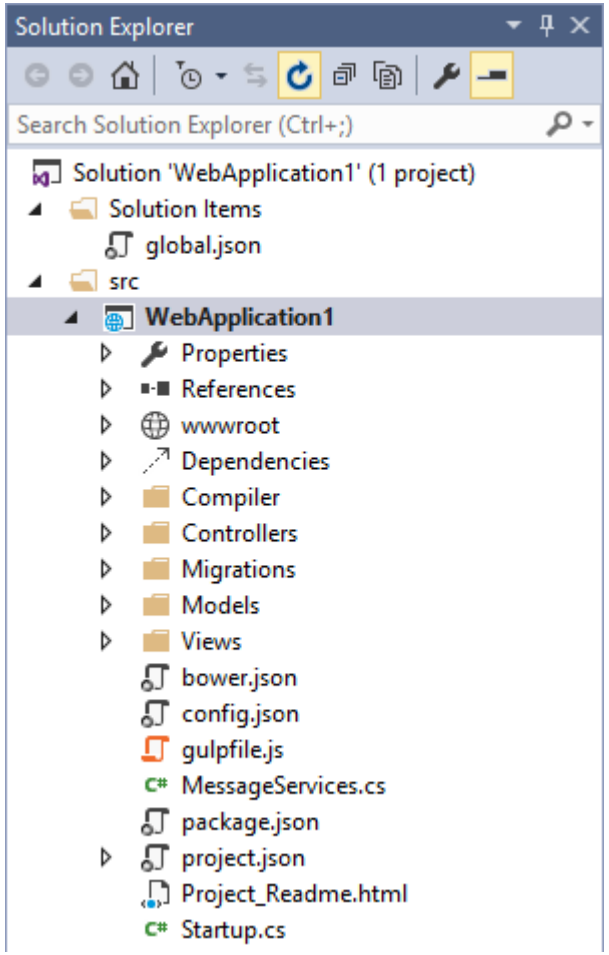
Client-Side Task Runners



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

More in “Client Side” section

Configuring the Application

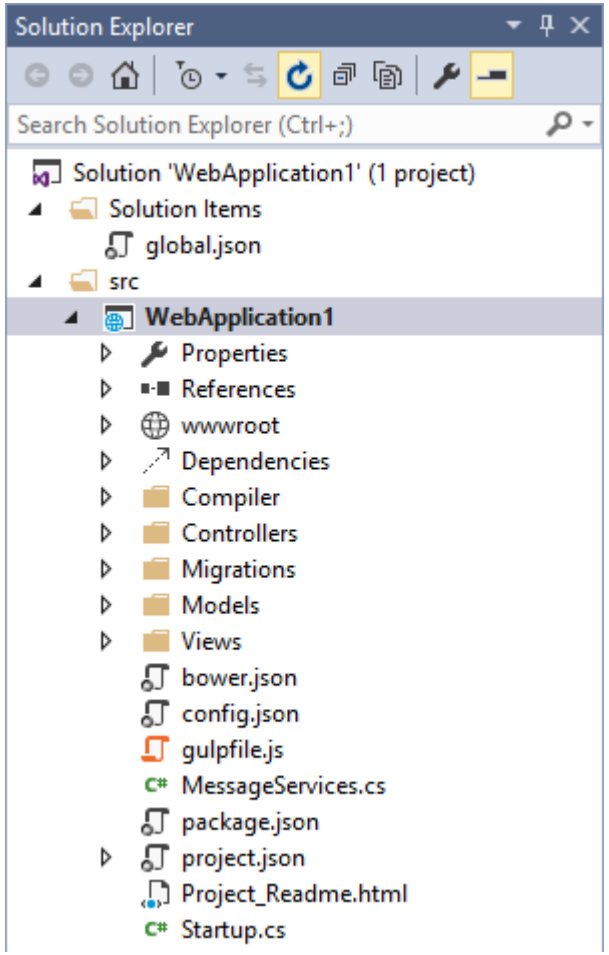


Web.Config -> *.json

- 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 5 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 5 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"
 },
```

HttpContext

HttpContext.Current doesn't exist anymore

Two Ways to Access:

- IHttpContextAccessor with DI
- MVC Controller base class property

Goodbye, HttpModules

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*/ );
}
```

So Long, HttpHandlers

Return custom response for specific path

MVC and Web API have this covered

Tag Helpers

New in MVC 6

Html Helpers

Register.cshtml | HomeController.cs | Contact.cshtml

```
@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizonta
{
    @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>

<img src="sample.png" alt="img" asp-append-version="true" />

<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>
```

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(EmailSender 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

Recap

Versions & History of ASP.NET

ASP.NET Command Line

Project Structure

Pipeline

Tag Helpers

Dependency Injection Framework

Client-side Development

Interested in More?

www.asp.net/vnext

docs.asp.net

jtower.com/blog

If You Give $100, So Will I!

http://bit.ly/detroit-gives

"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, JTower.com

jtower@jtower.com

jtower.com/blog

jtowermi

GIVE @ http://bit.ly/detroit-gives