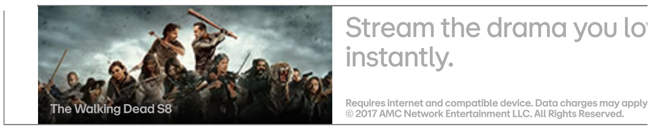
How to use Angular 4 with ASP.NET MVC 5


Stream the drama you lo
instantly.
The Walking Dead S8
Requires internet and compatible device. Data charges may apply
© 2017 AMC Network Entertainment LLC. All Rights Reserved.

Posted by Anuraj (https://plus.google.com/+AnurajP) on Saturday, November 25, 2017

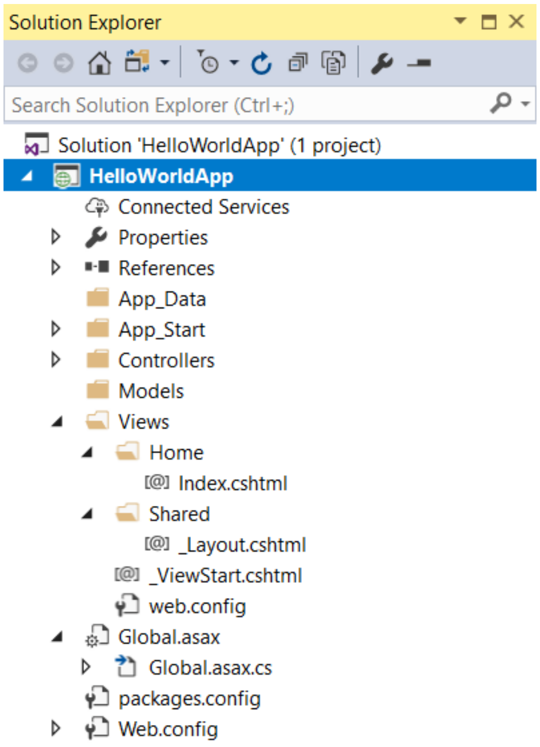
ASP.NET MVC (/tags#ASP.NET MVC) Angular (/tags#Angular)

This post is about how to use Angular 4 with ASP.NET MVC5. In one of my existing projects we were using Angular 1.x, due to some plugin compatibility issues, we had to migrate to latest version of Angular. We couldn't find any good article which talks about development and deployment aspects of Angular 4 with ASP.NET MVC.

The pre-requisites are

- npm (Node Package Manager) - I am using npm version 5.5.1
- Angular CLI - I am using angular cli version 1.5.3
- Visual Studio 2017 / 2015 - For developing ASP.NET MVC project.

First you need to create ASP.NET MVC project.



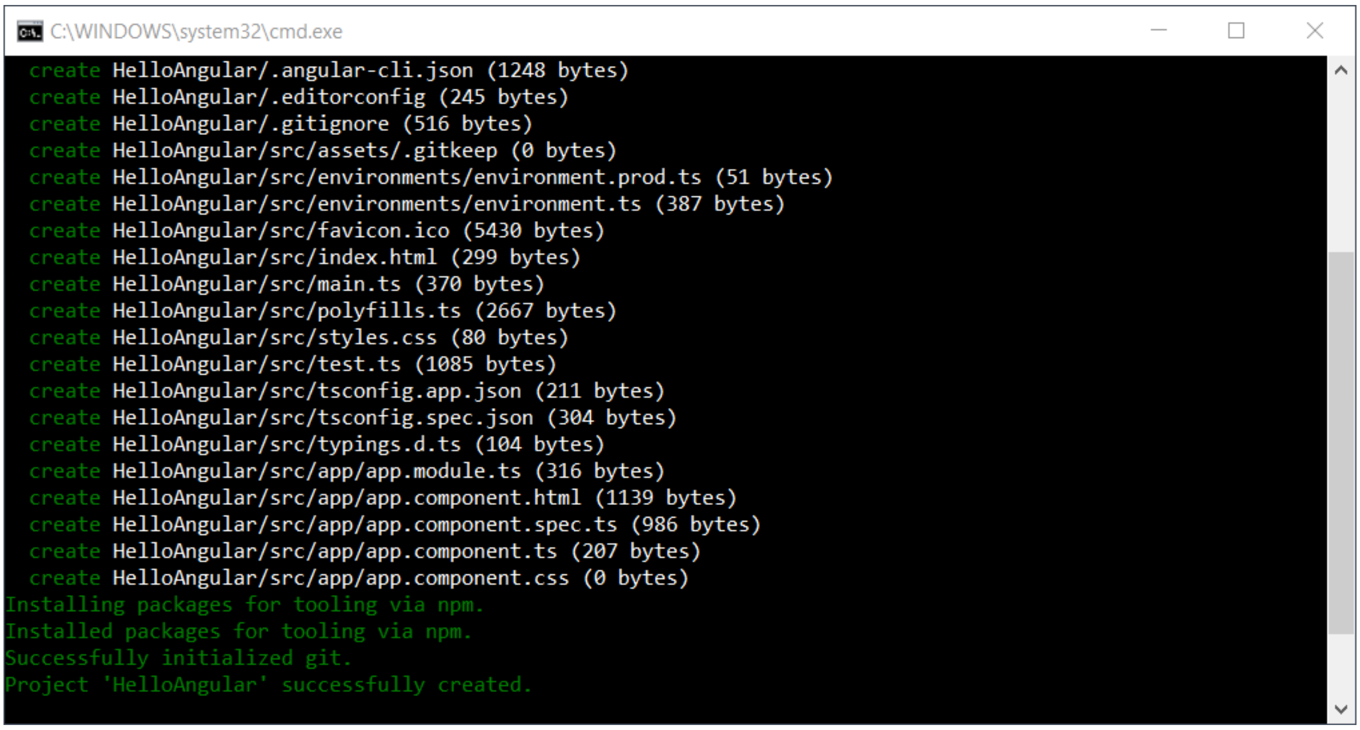
```
Solution 'HelloWorldApp' (1 project)
  HelloWorldApp
    Connected Services
    Properties
    References
    App_Data
    App_Start
    Controllers
    Models
    Views
      Home
        Index.cshtml
      Shared
        _Layout.cshtml
      _ViewStart.cshtml
      web.config
    Global.asax
      Global.asax.cs
    packages.config
    Web.config
```

Once you created the project, using Angular CLI you need to create Angular project. You need to do it in the root folder. And you need to execute following commands to create an Angular project.

```
ng new HelloAngular
```

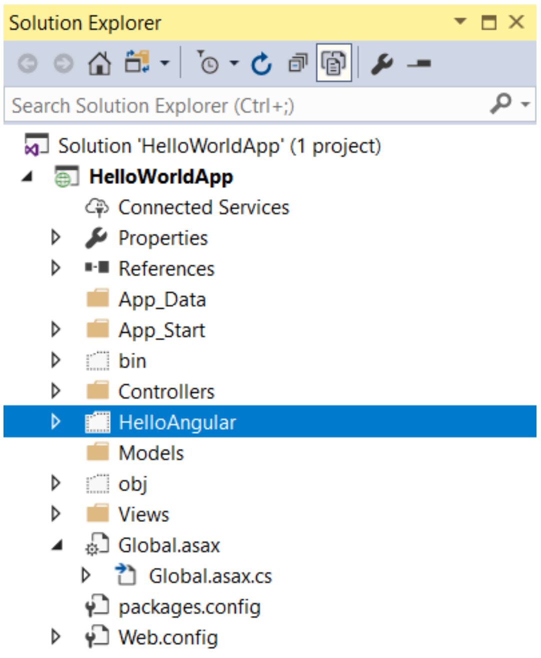
This will create a folder with name `HelloAngular`. And it contains all the required files for Angular project. Here is the folder structure after creating the angular project.

↑



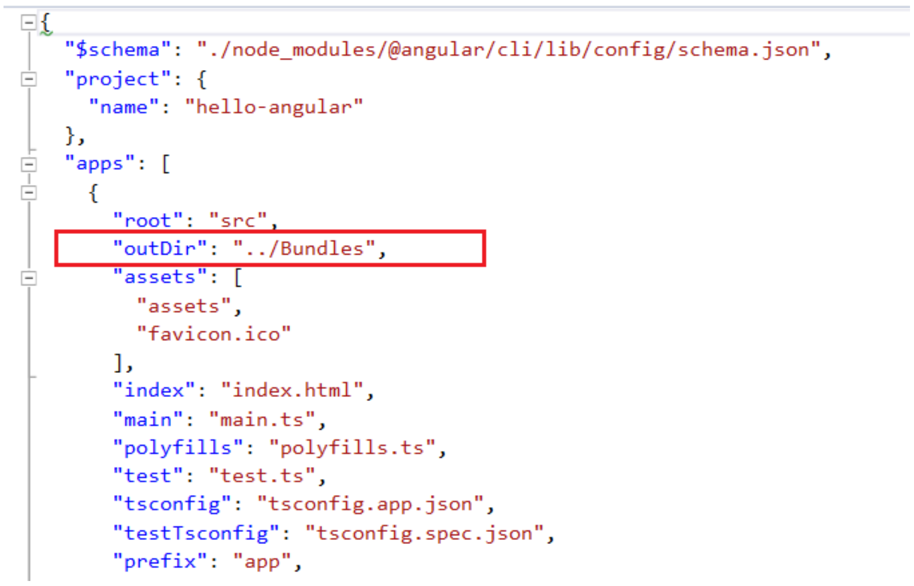
```
  create HelloAngular/.angular-cli.json (1248 bytes)
  create HelloAngular/.editorconfig (245 bytes)
  create HelloAngular/.gitignore (516 bytes)
  create HelloAngular/src/assets/.gitkeep (0 bytes)
  create HelloAngular/src/environments/environment.prod.ts (51 bytes)
  create HelloAngular/src/environments/environment.ts (387 bytes)
  create HelloAngular/src/favicon.ico (5430 bytes)
  create HelloAngular/src/index.html (299 bytes)
  create HelloAngular/src/main.ts (370 bytes)
  create HelloAngular/src/polyfills.ts (2667 bytes)
  create HelloAngular/src/styles.css (80 bytes)
  create HelloAngular/src/test.ts (1085 bytes)
  create HelloAngular/src/tsconfig.app.json (211 bytes)
  create HelloAngular/src/tsconfig.spec.json (304 bytes)
  create HelloAngular/src/typings.d.ts (104 bytes)
  create HelloAngular/src/app/app.module.ts (316 bytes)
  create HelloAngular/src/app/app.component.html (1139 bytes)
  create HelloAngular/src/app/app.component.spec.ts (986 bytes)
  create HelloAngular/src/app/app.component.ts (207 bytes)
  create HelloAngular/src/app/app.component.css (0 bytes)
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Successfully initialized git.
Project 'HelloAngular' successfully created.
```

Here is the updated folder structure after creating the Angular project.



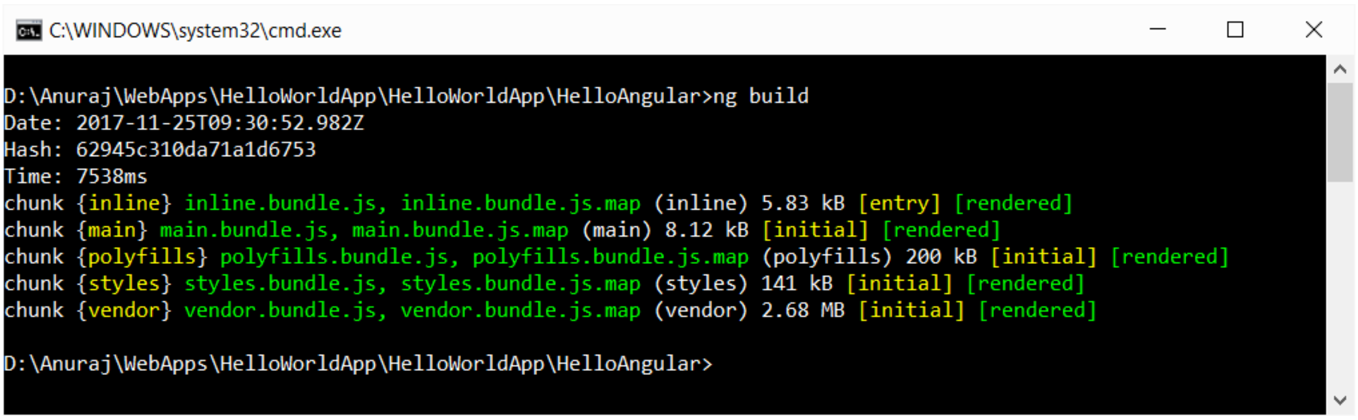
Unlike earlier versions of Angular, you need to reference the generated files with Angular CLI. You can generate the build files using `ng build` command. Before executing the ng build command, you need to configure your output folder, the scripts will be generated to the configured folder, by default it will be `dist`. I am configuring it to `../Bundles` folder.

↑



```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "project": {
    "name": "hello-angular"
  },
  "apps": [
    {
      "root": "src",
      "outDir": "../Bundles",
      "assets": [
        "assets",
        "favicon.ico"
      ],
      "index": "index.html",
      "main": "main.ts",
      "polyfills": "polyfills.ts",
      "test": "test.ts",
      "tsconfig": "tsconfig.app.json",
      "testTsconfig": "tsconfig.spec.json",
      "prefix": "app",
```

Next you need to run `ng build` command. Once you run the command, it will generate required files to the bundles folder.



C:\WINDOWS\system32\cmd.exe

```
D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>ng build
Date: 2017-11-25T09:30:52.982Z
Hash: 62945c310da71a1d6753
Time: 7538ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.12 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 200 kB [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 141 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.68 MB [initial] [rendered]

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>
```

Next you need to open `index.html` under bundles folder, you need to copy the script and style references from the file and add it to the _layout.cshtml file. I am using the ASP.NET MVC bundling and minification framework for this. Here is the code snippet for that.

```
public class BundleConfig
{
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/Script/Bundles")
                .Include(
                "~/bundles/inline.*",
                "~/bundles/polyfills.*",
                "~/bundles/scripts.*",
                "~/bundles/vendor.*",
                "~/bundles/main.*"));
    }
}
```

And here is my _layout.cshtml file.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello World - ASP.NET MVC</title>
</head>
<body>
    @RenderBody()
    @Scripts.Render("~/Script/Bundles")
</body>
</html>
```

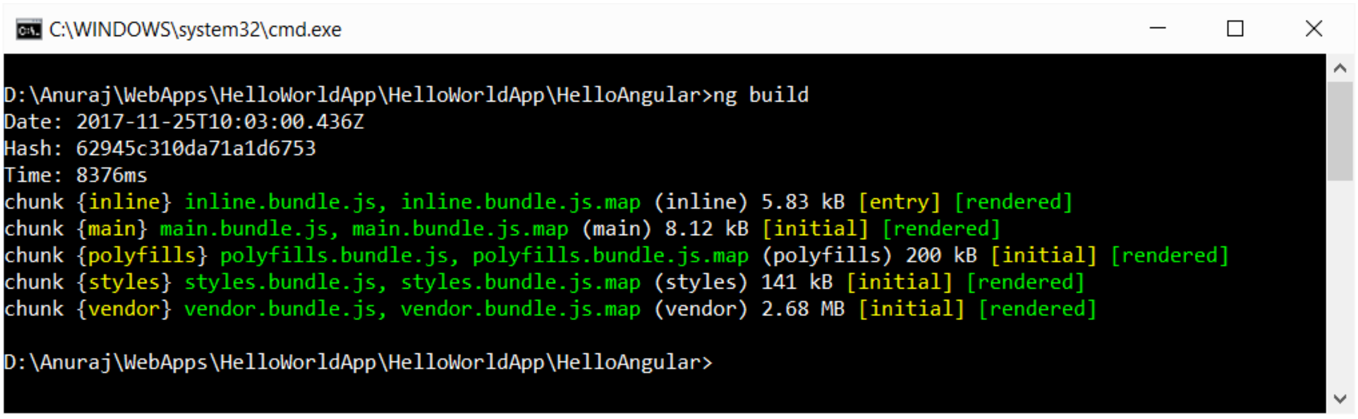
Now you have completed the initial infrastructure.

If you notice, I have removed the Bootstrap reference from _layout file. You can add it via `.angular-cli.json` file. First you need to install JQuery, Popper JS and Bootstrap 4 via `npm install` command. Once you installed it, you can modify the file like this.

↑

```
"prefix": "app",
"styles": [
  "styles.css",
  "../node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "../node_modules/jquery/dist/jquery.min.js",
  "../node_modules/popper.js/dist/umd/popper.min.js",
  "../node_modules/bootstrap/dist/js/bootstrap.min.js"
],
```

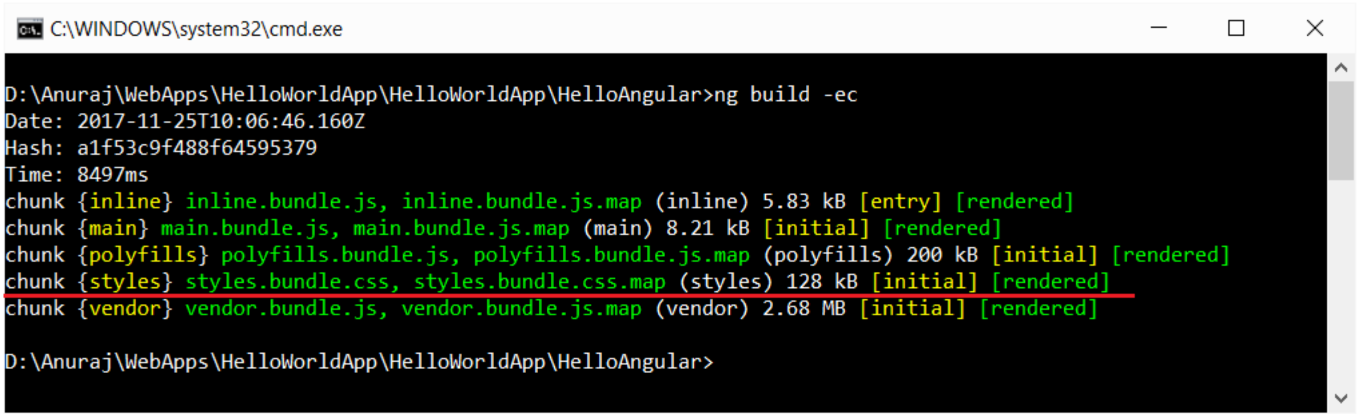
Again you can run the `ng build` command, which ideally should bundles the scripts and styles we configured in the `.angular-cli.json` file.



```
D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>ng build
Date: 2017-11-25T10:03:00.436Z
Hash: 62945c310da71a1d6753
Time: 8376ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.12 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 200 kB [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 141 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.68 MB [initial] [rendered]

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>
```

If notice, even though we added the style reference, `ng build` command didn't generated any css files. You need to provide `-ec` command line parameter to `ng build` command. This will generate the styles.



```
D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>ng build -ec
Date: 2017-11-25T10:06:46.160Z
Hash: a1f53c9f488f64595379
Time: 8497ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.21 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 200 kB [initial] [rendered]
chunk {styles} styles.bundle.css, styles.bundle.css.map (styles) 128 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.68 MB [initial] [rendered]

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>
```

Now you can modify your bundleconfig file like this, including the style reference.

```
bundles.Add(new StyleBundle("~/Content/Styles")
    .Include("~/bundles/styles.*"));
```

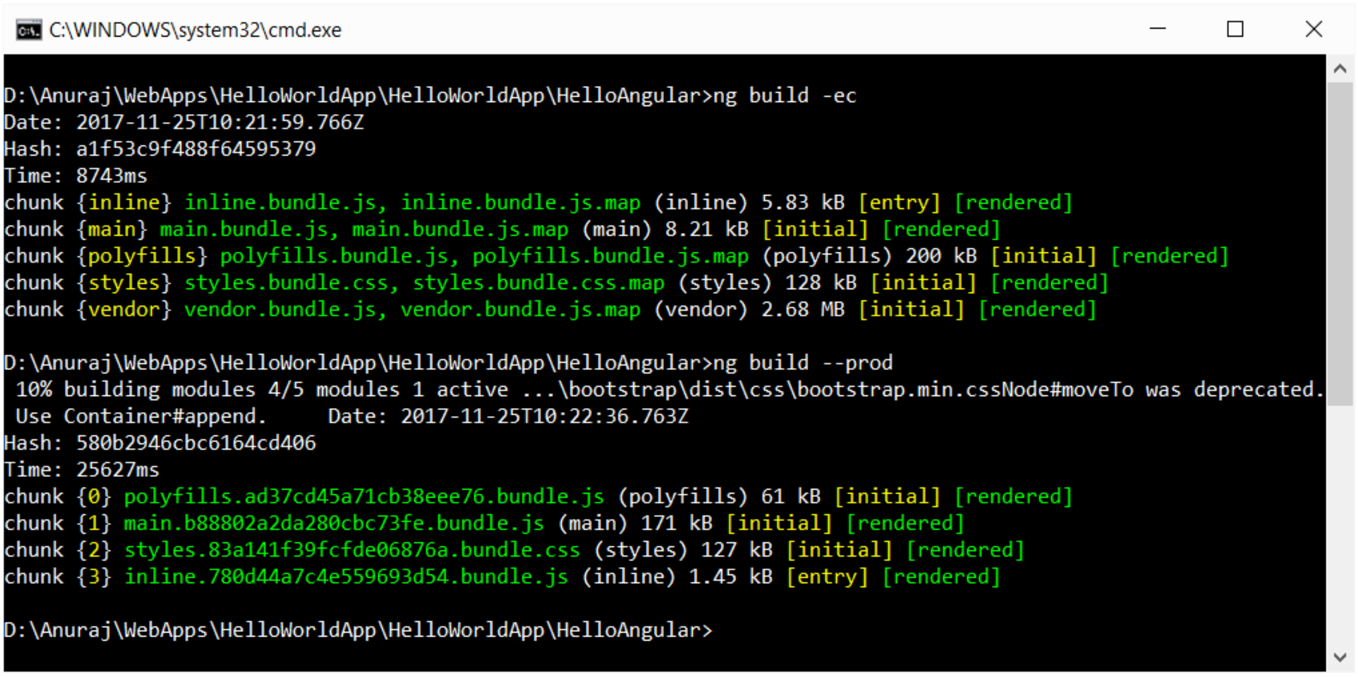
And _layout file like this.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hello World - ASP.NET MVC</title>
    @Styles.Render("~/Content/Styles")
</head>
<body>
    <main role="main" class="container">
        @RenderBody()
    </main>
    @Scripts.Render("~/Script/Bundles")
</body>
</html>
```

Now you can run the ASP.NET MVC application to view the results.

You can run the `ng build --prod` command to make your script file minified, which is required to deploy the application in server. You don't need `-ec` parameter to generate the styles. Also the `--prod` parameter will reduce the file size as well.

↑



```
C:\WINDOWS\system32\cmd.exe

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>ng build -ec
Date: 2017-11-25T10:21:59.766Z
Hash: a1f53c9f488f64595379
Time: 8743ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.21 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 200 kB [initial] [rendered]
chunk {styles} styles.bundle.css, styles.bundle.css.map (styles) 128 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.68 MB [initial] [rendered]

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>ng build --prod
 10% building modules 4/5 modules 1 active ...\bootstrap\dist\css\bootstrap.min.cssNode#moveTo was deprecated.
 Use Container#append.     Date: 2017-11-25T10:22:36.763Z
Hash: 580b2946cbc6164cd406
Time: 25627ms
chunk {0} polyfills.ad37cd45a71cb38eee76.bundle.js (polyfills) 61 kB [initial] [rendered]
chunk {1} main.b88802a2da280cbc73fe.bundle.js (main) 171 kB [initial] [rendered]
chunk {2} styles.83a141f39fcfde06876a.bundle.css (styles) 127 kB [initial] [rendered]
chunk {3} inline.780d44a7c4e559693d54.bundle.js (inline) 1.45 kB [entry] [rendered]

D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\HelloAngular>
```

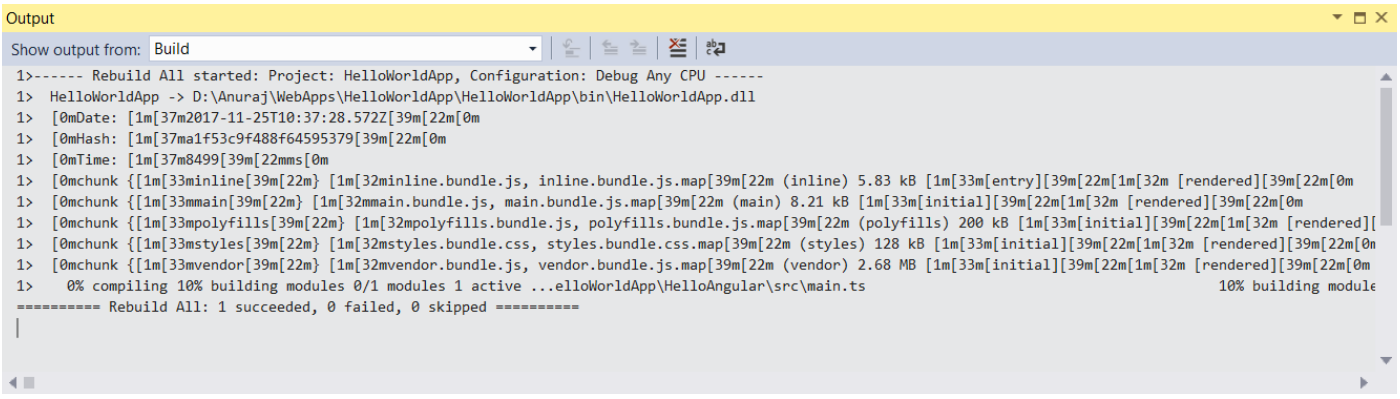
You can find the difference of file sizes.

Next you can modify the project file to automate the deployment. You need to edit your project file. For that first you need to unload the project by right clicking on the project node and select `Unload Project` option. Then select the `Edit HelloWorldApp.csproj` . Next add the following code inside the project element.

```
<Target Name="NgDebug" BeforeTargets="Build" Condition="'$(Configuration)' == 'Debug'">
  <Exec WorkingDirectory="$(ProjectDir)HelloAngular" Command="ng build -ec" />
</Target>
<Target Name="NgRelease" BeforeTargets="Build" Condition="'$(Configuration)' == 'Release'">
  <Exec WorkingDirectory="$(ProjectDir)HelloAngular" Command="ng build --prod" />
</Target>
```

The above code will execute `ng build -ec` and `ng build --prod` commands based on the Visual Studio configuration. If the configuration is Debug, I will be executing the `ng build -ec` command and for Release configuration I am executing the `ng build --prod` command.

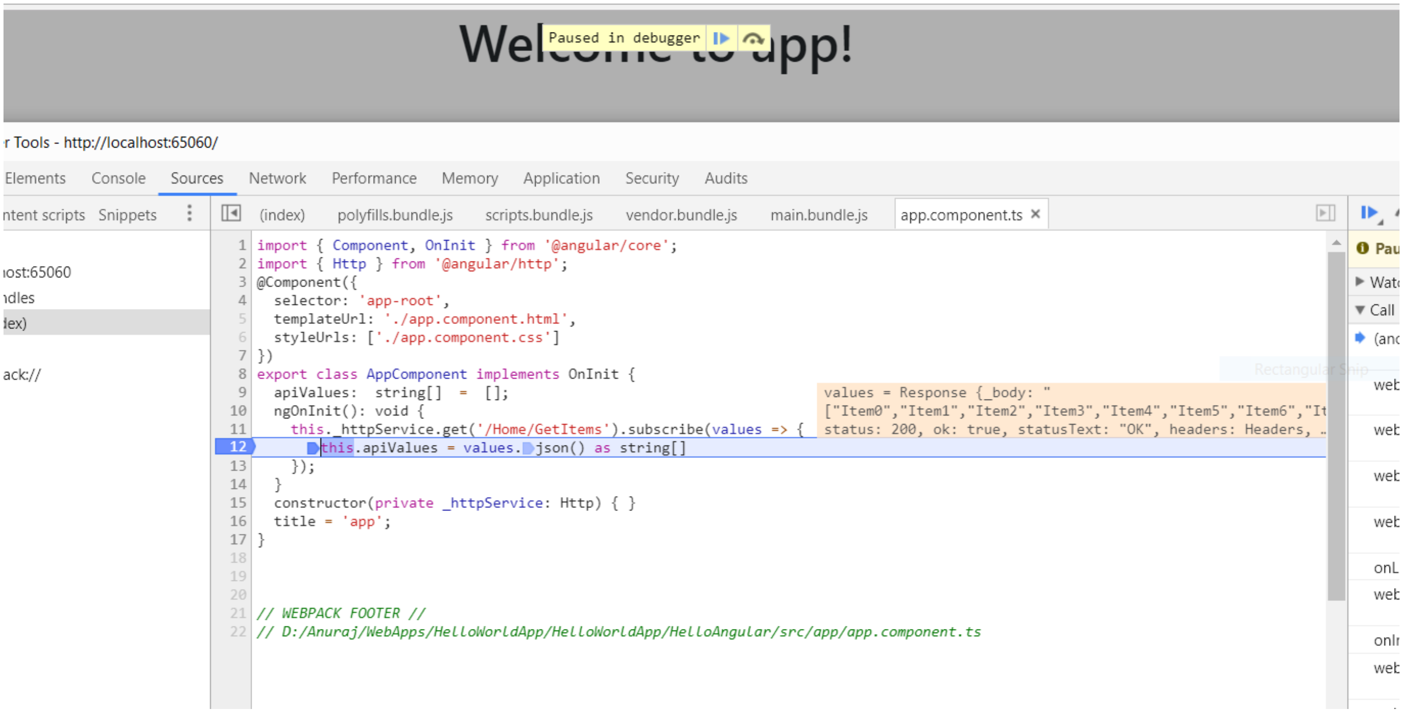
Next reload the project and build it. You will be able to see the `ng build` command output in the output window of Visual Studio.



```
1>------ Rebuild All started: Project: HelloWorldApp, Configuration: Debug Any CPU ------
1>  HelloWorldApp -> D:\Anuraj\WebApps\HelloWorldApp\HelloWorldApp\bin\HelloWorldApp.dll
1>  [0mDate: [1m[37m2017-11-25T10:37:28.572Z[39m[22m[0m
1>  [0mHash: [1m[37ma1f53c9f488f64595379[39m[22m[0m
1>  [0mTime: [1m[37m8499[39m[22mms[0m
1>  [0mchunk {[1m[33minline[39m[22m} [1m[32minline.bundle.js, inline.bundle.js.map[39m[22m (inline) 5.83 kB [1m[33m[entry][39m[22m[1m[32m [rendered][39m[22m[0m
1>  [0mchunk {[1m[33mmain[39m[22m} [1m[32mmain.bundle.js, main.bundle.js.map[39m[22m (main) 8.21 kB [1m[33m[initial][39m[22m[1m[32m [rendered][39m[22m[0m
1>  [0mchunk {[1m[33mpolyfills[39m[22m} [1m[32mpolyfills.bundle.js, polyfills.bundle.js.map[39m[22m (polyfills) 200 kB [1m[33m[initial][39m[22m[1m[32m [rendered][
1>  [0mchunk {[1m[33mstyles[39m[22m} [1m[32mstyles.bundle.css, styles.bundle.css.map[39m[22m (styles) 128 kB [1m[33m[initial][39m[22m[1m[32m [rendered][39m[22m[0n
1>  [0mchunk {[1m[33mvendor[39m[22m} [1m[32mvendor.bundle.js, vendor.bundle.js.map[39m[22m (vendor) 2.68 MB [1m[33m[initial][39m[22m[1m[32m [rendered][39m[22m[0m
1>    0% compiling 10% building modules 0/1 modules 1 active ...elloWorldApp\HelloAngular\src\main.ts                          10% building module
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

That's it. Now you can add modules and components to angular application and you can debug it with chrome developer tools.

↑



```
import { Component, OnInit } from '@angular/core';
import { Http } from '@angular/http';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  apiValues:  string[]  =  [];
  ngOnInit(): void {
    this._httpService.get('/Home/GetItems').subscribe(values => {     values = Response {_body: "
                                                                       ["Item0","Item1","Item2","Item3","Item4","Item5","Item6","It
                                                                       status: 200, ok: true, statusText: "OK", headers: Headers, …
      this.apiValues = values.json() as string[]
    });
  }
  constructor(private _httpService: Http) { }
  title = 'app';
}



// WEBPACK FOOTER //
// D:/Anuraj/WebApps/HelloWorldApp/HelloWorldApp/HelloAngular/src/app/app.component.ts
```

Happy Programming :)

Did you like this article? Share it with your friends

Facebook (https://www.facebook.com/sharer/sharer.php?u=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

Twitter (https://twitter.com/home?status=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

Google+ (https://plus.google.com/share?url=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

LinkedIn (https://www.linkedin.com/shareArticle?mini=true&url=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

Reddit (https://reddit.com/submit?url=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

StumbleUpon (https://www.stumbleupon.com/submit?url=https://dotnetthoughts.net/how-to-use-angular4-wth-aspnet-mvc/)

« Using LESS CSS with ASP.NET Core (/using-less-css-with-aspnet-core/)   CI build for an ASP.NET Core app » (/ci-build-for-an-aspnet-core-app/)

↑

13 Comments **dotnetthoughts** 1 **Login**

Recommend Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Umar Kashmiri • 11 days ago

Finally we made it working by setting --deploy-url along with ngbuild. I was deploying this as sub applicaiton in IIS due to which angular was unable to find the chunk paths of the deployed folder.
--deploy-url = "http://xyz.com/subapplicati..."

• Reply • Share ›

Anuraj P Mod Umar Kashmiri • 11 days ago

Happy to see it is working :)

1 • Reply • Share ›

Umar Kashmiri • 12 days ago

I just tested your approach and it does not work with lazy loading of modules. Any suggestion? how angular lazy loading call chunks of js internally?

• Reply • Share ›

Anuraj P Mod Umar Kashmiri • 12 days ago

That's strange. I didn't tested with lazy loading modules. What I did is moving the script reference from ng build output folder index.html file to MVC bundling engine. Can you try the same stuff?

• Reply • Share ›

Umar Kashmiri Anuraj P • 11 days ago

I did try actually... It seems like, in case of lazy loading angular tries to fetch chunk files from the server and either path is not available or MVC routes are restricting that.. Not sure how angular internally load modules in lazy way... need to dig more.

• Reply • Share ›

Umar Kashmiri Anuraj P • 12 days ago

I had tried the same but unfortunately angular is trying to access lazy module chunks from somewhere. May be remotely from server and mvc is restricting the path..just a guess

• Reply • Share ›

Anuraj P Mod Umar Kashmiri • 11 days ago

Can you put some minimal lazy loading example code? I will look into it and update.

• Reply • Share ›

Umar Kashmiri Anuraj P • 11 days ago

https://github.com/umarkash...

I had created angular seed project here but this was done using asp.net core.. .
you can grab angular cli project only from here.

• Reply • Share ›

Umar Kashmiri • 14 days ago

Will this create one bundle file for all your scripts generate by angular cli?

• Reply • Share ›

Anuraj P Mod Umar Kashmiri • 13 days ago

No Angular CLI generates multiple files, if you use aspnet bundling and minification, you can create one single script file.

• Reply • Share ›

Umar Kashmiri Anuraj P • 12 days ago

I don't have intention to create one single file as this is ruin the
purpose. I was only wondering if i have lazy loading of modules with
routing, angular cli by default creates chunks of the files by default.
Not sure whether this approach works with lazy loading.. Have you tested
that?
Nice article though

• Reply • Share ›

boopathi • 18 days ago

Thank you. This is helped us a lot

• Reply • Share ›

↑