Angular 4.x and ASP.Net Core 1.1 In Visual Stu

Angular-CLI + Angular 4.x + ASP.Net Core 1.1 in Visual Studio 2017

This document was put together in early June 2017 as part of a research project to determine if/how Microsoft ASP.Net Core and Angular could be used together to create a moc power of Angular 4.x with the power of Microsoft middleware (ASP.Net Core Web Application) without using all of the "heavier" pieces of Microsoft ASP.Net MVC 5.

Using ASP.Net Core instead of the full ASP.Net MVC allows the "middle" to exist on non-Windows servers.

The research started with an article written a long time ago and augmented with information that would allow someone to loosely follow the instructions in the original article, au was written near the end of March 2017 (yes, the tools change **that fast**).

The tools that I ended up using for this project were:

- NodeJS 7.10.0
- npm 5.0.3
- TypeScript 2.3.4
- Webpack 2.6.1
- Angular-CLI 1.1.0
- Angular 4.1.3
- Visual Studio 2017 Enterprise (fully patched with all 7 available patches at the time of this writing)

Original Article

The original article I used to get started is located at:

http://www.sparkhound.com/learn/blog/setting-up-an-asp-net-core-project-with-angular-2-utilizing-angular-cli

Changes I made since the article was first written

1. Before beginning the steps called for in the article, you must make sure you have installed *NodeJS* and *npm*. I like to install *NodeJS* into directory `c:\nodeJS` **and** put in fairly

2. I like to keep the packages used in an Angular application up-to-date; but, it is painful to do so manually.
   Thankfully, if you install `npm-check-updates`, the task can be automated.
   In fact, in one of the later steps below, I actually use that package, via its short name, `ncu`, so if you have not already installed it, install it **globally** now via a PowerShell or Cc

   ```
   npm install -g npm-check-updates
   ```

3. After creating the new ASP.Net Core Web application called for in the original article, open it up in Visual Studio 2017.

4. Right-click on the project (*not the Solution*) and choose *Manage NuGet Packages...*.

5. Use the *Update* tab to update the ASP.Net Core packages to the latest editions.

6. When creating the new Angular-CLI project in a temp directory, make sure you specify the following options:

   - --skip-git
   - --skip-install
   - --routing

   example:

12. In the original article, it has you open up file `angular-cli.json`, but that has changed to `.angular-cli.json` (notice the leading period). You still need to change this:

    `"outdir": "dist",`

    to this:

    `"outdir": "wwwroot",`

13. In the original article, they do not have you modify file `package.json` to change the Angular-CLI build command, but I like to use *cache busting* and other compilation *extras* does not catch. Therefore, I recommend editing your `package.json` to change this:

    ```
    "build": "ng build",
    ```

    to this:

    ```
    "build": "ng build -ec -vc --progress --prod",
    ```

14. If you decide to try out the *bonus* portion of the original article, one **extremely important** step that the original article fails to mention is the need to update file `app.module`

- Add the following import to the top of the file after the existing imports of `BrowserModule` and `NgModule`:

  ```
  import { HttpModule } from '@angular/http';
  ```

- Add `HttpModule` to the `imports:` array:

  ```
  imports: [
    BrowserModule,
    HttpModule,
    AppRoutingModule
  ],
  ```

15. The bonus section of the original article was written prior to the release of Angular-CLI 1.1.0, which puts a lot of additional HTML markup into file `app.component.html`, so wh bonus section of the article, you should surround them in their own individual `<div></div>`:

    ```
    <div>
      <button type="button" (click)="onClick()">Test!</button>
    </div>

    <div>{{successMessage}}</div>
    ```

16. (optional) Publish to Azure by:

    1. right-clicking on the project in *Solution Explorer* inside Visual Studio 2017.
    2. choose *Publish*
    3. Fill in the information based on your Azure subscription. Hint: for the url of the web site, I like to remove the numbers that the publishing tool offers as a suffix (to make as shown below:

       ```
       http://fpmangularcliaspnetcore.azurewebsites.net
       ```