Visual Studio 2015 QuickStart

{@a top}

Some developers prefer Visual Studio as their Integrated Development Environment (IDE).

This cookbook describes the steps required to set up and use the Angular QuickStart files in **Visual Studio 2015 within an ASP.NET 4.x project**.

There is no *live example* for this cookbook because it describes Visual Studio, not the QuickStart application itself.

{@a asp-net-4}

ASP.NET 4.x Project

To set up the QuickStart files with an **ASP.NET 4.x project** in Visual Studio 2015, follow these steps:

If you prefer a `File | New Project` experience and are using **ASP.NET Core**, then consider the _experimental_ ASP.NET Core + Angular template for Visual Studio 2015. Note that the resulting code does not map to the docs. Adjust accordingly.

Prerequisite: Node.js

Install **Node.js® and npm** if they are not already on your machine.

Verify that you are running node version `4.6.x` or greater, and npm `3.x.x` or greater by running `node -v` and `npm -v` in a terminal window. Older versions produce errors.

Prerequisite: Visual Studio 2015 Update 3

The minimum requirement for developing Angular applications with Visual Studio is Update 3. Earlier versions do not follow the best practices for developing applications with TypeScript. To view your version of Visual Studio 2015, go to `Help | About Visual Studio`.

If you don't have it, install **Visual Studio 2015 Update 3**. Or use `Tools | Extensions and Updates` to update to Update 3 directly from Visual Studio 2015.

Prerequisite: Configure External Web tools

Configure Visual Studio to use the global external web tools instead of the tools that ship with Visual Studio:

- Open the **Options** dialog with `Tools` | `Options`.
- In the tree on the left, select `Projects and Solutions` | `External Web Tools`.
- On the right, move the `$(PATH)` entry above the `$(DevEnvDir)` entries. This tells Visual Studio to use the external tools (such as npm) found in the global path before using its own version of the external tools.
- Click OK to close the dialog.
- Restart Visual Studio for this change to take effect.

Visual Studio now looks first for external tools in the current workspace and if it doesn't find them, it looks in the global path. If Visual Studio doesn't find them in either location, it will use its own versions of the tools.

Prerequisite: Install TypeScript 2.2 for Visual Studio 2015

While Visual Studio Update 3 ships with TypeScript support out of the box, it currently doesn't ship with TypeScript 2.2, which you need to develop Angular applications.

To install TypeScript 2.2:

- Download and install **TypeScript 2.2 for Visual Studio 2015**
- OR install it with npm: `npm install -g typescript@2.2`.

You can find out more about TypeScript 2.2 support in Visual studio **here**.

At this point, Visual Studio is ready. It's a good idea to close Visual Studio and restart it to make sure everything is clean.

Step 1: Download the QuickStart files

Download the QuickStart source from GitHub. If you downloaded as a zip file, extract the files.

Step 2: Create the Visual Studio ASP.NET project

Create the ASP.NET 4.x project in the usual way as follows:

- In Visual Studio, select `File` | `New` | `Project` from the menu.

- In the template tree, select `Templates` | `Visual C#` (or `Visual Basic`) | `Web`.
- Select the `ASP.NET Web Application` template, give the project a name, and click OK.
- Select the desired ASP.NET 4.5.2 template and click OK.

This cookbook uses the `Empty` template with no added folders, no authentication, and no hosting. Pick the template and options appropriate for your project.

Step 3: Copy the QuickStart files into the ASP.NET project folder

Copy the QuickStart files you downloaded from GitHub into the folder containing the `.csproj` file. Include the files in the Visual Studio project as follows:

- Click the `Show All Files` button in Solution Explorer to reveal all of the hidden files in the project.
- Right-click on each folder/file to be included in the project and select `Include in Project`. Minimally, include the following folder/files:
    - src/app folder (answer *No* if asked to search for TypeScript Typings)
    - src/styles.css
    - src/index.html
    - package.json
    - src/tsconfig.json

Step 4: Restore the required packages

Restore the packages required for an Angular application as follows:

- Right-click on the `package.json` file in Solution Explorer and select `Restore Packages`. This uses `npm` to install all of the packages defined in the `package.json` file. It may take some time.
- If desired, open the Output window (`View` | `Output`) to watch the npm commands execute.
- Ignore the warnings.
- When the restore is finished, a message in the bottom message bar of Visual Studio should say: `Installing packages complete`. Be patient. This could take a while.
- Click the `Refresh` icon in Solution Explorer.
- **Do not** include the `node_modules` folder in the project. Let it be a hidden project folder.

Step 5: Build and run the app

First, ensure that `src/index.html` is set as the start page. Right-click `index.html` in Solution Explorer and select option `Set As Start Page`.

To run in VS with F5

Most Visual Studio developers like to press the F5 key and see the IIS server come up. To use the IIS server with the QuickStart app, you must make the following three changes.

1. In `index.html`, change base href from `<base href="/">` to `<base href="/src/">`.
2. Also in `index.html`, change the scripts to use `/node_modules` with a slash instead of `node_modules` without the slash.
3. In `src/systemjs.config.js`, near the top of the file, change the npm `path` to `/node_modules/` with a slash.

After these changes, `npm start` no longer works. You must choose to configure _either_ for F5 with IIS _or_ for `npm start` with the lite-server.

For apps that use routing

If your app uses routing, you need to teach the server to always return `index.html` when the user asks for an HTML page for reasons explained in the Deployment guide.

Everything seems fine while you move about *within* the app. But you'll see the problem right away if you refresh the browser or paste a link to an app page (called a "deep link") into the browser address bar.

You'll most likely get a *404 - Page Not Found* response from the server for any address other than `/` or `/index.html`.

You have to configure the server to return `index.html` for requests to these "unknown" pages. The `lite-server` development server does out-of-the-box. If you've switched over to F5 and IIS, you have to configure IIS to do it. This section walks through the steps to adapt the QuickStart application.

Configure IIS rewrite rules

Visual Studio ships with IIS Express, which has the rewrite module baked in. However, if you're using regular IIS you'll have to install the rewrite module.

Tell Visual Studio how to handle requests for route app pages by adding these rewrite rules near the bottom of the `web.config`:

<system.webServer> <rewrite> <rules> <rule name="Angular Routes" stopProcessing="true"> <match url=".*" /> <conditions logicalGrouping="MatchAll"> <add input="{REQUEST*FILENAME}" matchType="IsFile"*

*negate="true" /> <add input="{REQUEST*FILENAME}" matchType="IsDirectory" negate="true" /> </conditions> <action type="Rewrite" url="/src/" /> </rule> </rules> </rewrite> </system.webServer>

The match url, ``, will rewrite every request. You'll have to adjust this if you want some requests to get through, such as web API requests. The URL in `` should match the base href in `index.html`.

Build and launch the app with debugger by clicking the **Run** button or by pressing `F5`.

It's faster to run without the debugger by pressing `Ctrl-F5`.

The default browser opens and displays the QuickStart sample application.

Try editing any of the project files. Save and refresh the browser to see the changes.