How to Add Angular Schematic Projects to the Angular Workspace (or Nx)

The Angular development environment since version 6 is a `Workspace` by default. The workspace allows for a `monorepo` that can contain multiple `application` projects as well as one ore more shared `library` projects. But the question is, "What kind of project is a Schematic?".

Is a Schematic project a `library`? It is not shared by `application` projects and referenced as a module import. It is not an application - it is not hosted, it does not load and use other Angular modules. An Angular `Schematic` project appears to be something of its own type for the following reasons/characteristics:

- tool
- utility
- used during development as a `devDependency`
    - create/new items in a project
    - update existing items in a project
- not an application `dependency`
- can be installed globally

A Schematic project needs a development environment that has the capability to not only develop a schematic, test a schematic, but also to be used by either `library` or `application` project types within an Angular Workspace. So, let's get started.

Our goals are:

- add a schematic project to an Nx Angular Worksapce environment
- build a schematic to an output directory in preparation of publishing to a package repository
    - not mixing build output with source files
    - include `README.md` for the output.

Schematic Tooling

Install the `schematics-cli` from the `@angular-devkit` using npm. This cli has schematics for creating schematics. It contains (2) schematics in the collection that will allow you to create:

- blank: a blank schematic with no implementation
- schematics: a sample of (3) schematics in a collection

These are good to start with to get familiar with the structure and concepts of schematics in general.

```
npm install global @angular-devkit/schematics-cli
schematics --help
schematics --list-schematics
```

Getting Started

Let's use the Nrwl.io Nx extension and schematics for our workspace.

```
npm install -g @nrwl/schematics @nrwl/nx
```

Create a new workspace environment using the Nrwl.io Nx extension - which is a set of schematics that use and provide additional features to the core Angular Schematics.

Use the following command to create a new Nx workspace.

```
create-nx-workspace learn-schematics --npm-scope=angularlicious
```

Create a Schematic

Create a new folder in the root of the workspace called `schematics`. This is a container for a new project type that is not currently supported or has a project type when using the `angular.json` configuration file.

```
mkdir schematics
```

Use the terminal and navigate to the `schematics` folder. Use the command to create a sample

```
schematics schematic --name=learn-schematics
```

The new schematics project has a `tsconfig.json` file to configure the build for Typescript. Update the configuration to include an `outDir` property that points the output to a `dist` folder.

```
"outDir": "./../../dist/schematics/getting-started",
```

In order to output the build of a specific schematic project to an output folder, update the build scripts of the workspace to include:

```
"build:schematics": "npm run build-schematic:getting-started",
"build-schematic:getting-started": "tsc -p ./schematics/getting-
started/tsconfig.json && npm run copy-schematic:getting-started-templates && npm
run copy-schematic:getting-started-types && npm run copy-schematic:getting-
started-package && npm run copy-schematic:getting-started-collection  && npm run
copy-schematic:getting-started-readme",
"copy-schematic:getting-started-templates": "sync-glob -d false
'schematics/getting-started/src/**/*/files/*' dist/schematics/getting-started/",
"copy-schematic:getting-started-types": "sync-glob -d false 'schematics/getting-
started/src/**/*/schema.*' dist/schematics/getting-started/",
"copy-schematic:getting-started-package": "sync-glob -d false 'schematics/getting-
```

```
started/package.json' dist/schematics/getting-started",
"copy-schematic:getting-started-collection": "sync-glob -d false
'schematics/getting-started/src/collection.json' dist/schematics/getting-started",
"copy-schematic:getting-started-readme": "sync-glob -d false 'schematics/getting-
started/README.md' dist/schematics/getting-started",
```

The `package.json` file in the `Schematic` project - contains a configuration that points to the `collection.json` file for the specified schematic.

- remove the `scripts` section from the `package.json` file
- update the `depdendencies` section to `peerDependencies`
- remove `src` from the `schematics` property:
    - change `"schematics": "./src/collection.json",`
    - to `"schematics": "./collection.json",`

```
{
  "name": "@angularlicious/getting-started",
  "version": "0.0.0",
  "description": "A schematics",
  "keywords": [
    "schematics"
  ],
  "author": "",
  "license": "MIT",
  "schematics": "./src/collection.json",
  "peerDependencies": {
    "@angular-devkit/core": "^7.0.5",
    "@angular-devkit/schematics": "^7.0.5",
    "@types/jasmine": "^2.6.0",
    "@types/node": "^8.0.31",
    "jasmine": "^2.8.0",
    "typescript": "~3.1.6"
  }
}
```

Update the root `package.json` file in the workspace. Add the following (2) items to the `dependency` section.

```
"@angular-devkit/core": "^7.0.5",
"@angular-devkit/schematics": "^7.0.5",
```

Build Schematic

Use the terminal and run the following command to build the schematics project. The output will be in the `dist` folder as defined in the `outDir` setting in the `tsconfig.json` file.

```
npm run build:schematics
```

Link the Schematic

Typically, you would publish your schematic to a package repository so it can be used like other schematics - using a `npm install -g <YOUR-SCHEMATIC-NAME-HERE>`.

However, in our local workspace development environment, use the `npm link` command to load the build output into the `node_modules`.

```
npm link ./dist/schematics/getting-started
```

Attempting to link to the ouput of the build, will throw an error. It appears that we are missing some required artifacts for the schematic - the output is missing the `package.json` file.

```
5 verbose stack Error: ENOENT: no such file or directory, open '.\learn-schematics-course\learn-schematics\dist\schematics\@angularlicious\getting-started\package.json'
```

There are actually a set of required files that a schematic project will need - not just the output of the transpile of the (Typescript, *.ts) files. We'll need:

- schema.* files
  - used to define the available `options` for the schematic (i.e., the inputs).
- templates (./files/*)
  - if any templates are defined, they will need to be available as part of the schematic project as an asset.
- package.json
  - all published packages require a `package.json` file with a unique name, version and other information.
- README.md: include documentation for the specified schematic
  - include details on how to use and test

Add the following package to the workspace.

```
npm install -D sync-glob
```

Using the Schematic

Now that the schematic is linked to the workspace (via `npm link <PATH_TO_THE_SCHEMATIC_BUILD_OUTPUT>`), you can use the schematic and target one of the schematics in the collection.

```
ng generate @angularlicious/getting-started:my-full-schematic --name="test"
```

The output of the command is (4) files based on the Schematic and the templates.

```
ng generate @angularlicious/getting-started:my-full-schematic --name="test"
    My Full Schematic: {"name":"test","index":1}
            My Other Schematic: {"option":true}
                        My Schematic: {"option":true}
CREATE hola (5 bytes)
CREATE allo (5 bytes)
CREATE test2 (34 bytes)
CREATE test1 (18 bytes)
```

Resources

Examples of Schematics in the Real World.

Angular Dev-Kit Schematics

- https://github.com/angular/angular-cli/tree/master/packages/schematics/schematics

Angular Schematics

- https://github.com/angular/angular-cli/tree/master/packages/schematics/angular