Add and Debug Schematic Projects in an Angular Workspace

It would be nice to have the ability to include `schematic` projects in your Angular Workspace, right? Many teams and developers are using a workspace and developing one or more applications that are also taking advantage of sharable library projects. We are also using the the `Angular CLI` in the workspace - so why not be able to develop, debug, test, use, and publish custom `schematic` projects from the `Workspace`.

> Previous to this post, I published an article about How to Debug an Angular Schematic using Visual Studio Code. It demonstrates how to setup debugging on the default schematic project configuration. I would love to see a new project type introduced to the Angular Workspace to support `schematic` projects. The category for this project type is `tools` - the Nrwl.io Nx workspace schematic already creates a `tools` folder for items like this.

The Objective:

- [ ] add a schematic to a workspace
- [ ] build the schematic for development, debugging, and testing
- [ ] debug the schematic
- [ ] run the schematic (default `--dry-run=true` mode)
- [ ] run the schematic in LIVE PERFORMANCE mode (`--dry-run=false`)

Configure the Workspace Package

A schematic project contains a special property in the `package.json` file. The `schematics` property is an indicator that the project type is a schematic. If you are using the `schematics` collection from the `@angular-devkit/schematics-cli`, each project will include a valid path to the `collection.json` file for the schematic.

```
"schematics": "./src/collection.json",
```

If the `schematics` property is missing from the project/workspace root `package.json`, you will get the following error:

```
An error occured:
Error: Package "." was found but does not support schematics.
```

If the `schematics` property contains an invalid path

```
An error occured:
Error: Collection "." cannot be resolved.
```

> Note: the incorrect spelling of *occurred* is from the CLI.

However, what do we do if we would like to add a `schematic` project type to our Angular Workspace? For now, we could update the workspace/root `package.json` to include the `schematics` property. You would have to manually point it to the specified collection that you are targeting for debugging.

```
"schematics": "./schematics/getting-started-with-debugging/src/collection.json",
```

Create a Workspace

Create a new workspace using Nrwl.io Nx schematics. You will require the `@nrwl/schematics` package installed before using the `create-nx-workspace` schematic.

```
npm install -g @nrwl/schematics
create-nx-workspace my-workspace-with-schematics
```

Tooling and Prerequisites

Before, we can create a schematic project using the `schematic-cli`, we'll need to make sure we have the following packages available to our development environment. Install the following packages using the `-g` to make them available globally.

> Note: the `@angular-devkit/schematics-cli` is ***also*** installed in the local workspace. This will make it easier to configure the launch configuration for debugging.

```
npm install -g @angular-devkit/schematics
npm install -g @angular-devkit/schematics-cli
npm install -D @angular-devkit/schematics-cli
```

Schematic Utilities

The Angular team has published a package that includes the `utility` items that we have been waiting for. Several awesome developers during the last year or so have created and maintained non-official utility packages for schematic development. However, now you can use the official utility package.

```
npm install -D @schematics/angular
```

> The current location of these [utilities](Angular Schematic Utilities) can be found here: https://github.com/angular/angular-cli/tree/master/packages/schematics/angular/utility

To use the utility functions, you will need to add `imports` from the specified utility item. You may need to peek into the utility items to determine where a specific function lives. For example:

```
import { parseName } from '@schematics/angular/utility/parse-name'
import { getWorkspace } from '@schematics/angular/utility/config'
import { buildDefaultPath } from '@schematics/angular/utility/project'
import { WorkspaceProject } from '@schematics/angular/utility/workspace-models';
```

Workspace Schematic Project

The tooling allows us to create a new `schematic` project using the `schematics` collection. Use the `blank` item in the collection to create a sample schematic collection with a single blank schematic.

- [ ] Create a new folder in the root of the workspace called `schematics`.
- [ ] Open a terminal to the `schematics` folder.
- [ ] Run the `schematics` command in a terminal.
  - [ ] `schematics blank --name=getting-started-with-debugging --dry-run`

> Note: remove the `--dry-run` option to create the project.

- [ ] Update the `collection.json` and the `index.ts` file of your new schematic project to use the following.

collection.json:

```
{
  "$schema": "../node_modules/@angular-devkit/schematics/collection-schema.json",
  "schematics": {
    "tree-debug": {
      "description": "A schematic to demonstrate how to debug and view [Tree]
details..",
      "factory": "./tree-debug"
    }
  }
}
```

index.ts:

```
import {
  Rule,
  SchematicContext,
  SchematicsException,
  Tree,
  apply,
  filter,
  mergeWith,
  move,
  noop,
  template,
  url,
  branchAndMerge,
} from '@angular-devkit/schematics';
import { strings } from '@angular-devkit/core';
import { parseName } from '@schematics/angular/utility/parse-name'
import { getWorkspace } from '@schematics/angular/utility/config'
import { buildDefaultPath } from '@schematics/angular/utility/project'
import { WorkspaceProject } from '@schematics/angular/utility/workspace-models';

/**
 * Use to setup the target path using the specified options [project].
 * @param host the current [Tree]
 * @param options the current [options]
 * @param context the [SchematicContext]
 */
export function setupOptions(host: Tree, options: any, context: SchematicContext)
{
  const workspace = getWorkspace(host);
  if (!options.project) {
    context.logger.error(`The [project] option is missing.`);
    throw new SchematicsException('Option (project) is required.');
  }
  context.logger.info (`Preparing to retrieve the project using:
${options.project}`);
  const project = <WorkspaceProject>workspace.projects[options.project];

  if (options.path === undefined) {
    context.logger.info(`Preparing to determine the target path.`);
    options.path = buildDefaultPath(project);
    context.logger.info(`The target path: ${options.path}`);
  }

  options.type = !!options.type ? `.${options.type}` : '';

  const parsedPath = parseName(options.path, options.name);
  options.name = parsedPath.name;
  options.path = parsedPath.path;

  context.logger.info(`Finished options setup.`);
  return host;
}

export default function (options: any): Rule {
  return (host: Tree, context: SchematicContext) => {

    setupOptions(host, options, context);

    // setup a variable [currentDateTime] programmatically --> used in template;
    options.currentDateTime = new Date(Date.now()).toUTCString();

    const templateSource = apply(url('./files'), [
      options.spec ? noop() : filter(path => !path.endsWith('.spec.ts')),
      template({
        ...strings,
        ...options,
      }),
      move(options.path),
    ]);

    return branchAndMerge(mergeWith(templateSource));
  };
}
```

- [ ] Add a new `files` folder in the `tree-debug` schematic.
- [ ] Add a new file in the folder called `debug.txt`.
- [ ] Update the `debug.txt` with the following template:

```
<% if (name) { %>
  Hello <%= name %>, I'm a debugged schematic at <%= currentDateTime %>.
<% } else { %>
  Why don't you give me your name with --name?
<% } %>
```

Build and Test the Schematic Project

The default project configuration for a schematic includes a build and test script for each schematic in the `package.json` folder. The path of the `tsconfig.json` and the `specification` test files are relative to the project's `package.json` file.

```
"build": "tsc -p tsconfig.json",
"test": "npm run build && jasmine src/**/*_spec.js"
```

However, since we are using an Angular Workspace project, we can create a new build script with the correct path.

- [ ] Add this to the workspace root `package.json` file.

```
"build-schematic:getting-started-with-debugging": "tsc -p ./schematics/getting-
started-with-debugging/tsconfig.json",
```

- [ ] Run the `build` or the `test` script for the target schematic project.

```
npm run build-schematic:getting-started-with-debugging
npm run test-schematic:getting-started-with-debugging
```

Debugging Schematics in Visual Studio Code within an Angular Workspace

Create a new `launch.json` configuration for the project. The type of configuration to add is `node.js - Launch Program`. When this specific configuration is selected for debugging, we'll need to target the node program. In this case, it is the `schematics.js` (from the `@angular-devkit/schematics-cli` package). The program we want is a Javascript file in the `bin` folder of the package.

I simplified the access to this program by installing the `@angular-devkit/schematics-lic` package locally in the project. The `program` property of the launch configuration requires the file to be a full path to the location of the program or Javascript file.

```
npm install -D @angular-devkit/schematics-cli
```

```
{
    "version": "0.2.0",
    "configurations": [
         {
            "type": "node",
            "request": "launch",
            "name": "With Debugging",
            "program": "${workspaceFolder}/node_modules/@angular-
devkit/schematics-cli/bin/schematics.js",
            "args": [
                ".:tree-debug"
            ],
            "outFiles": []
        }
    ]
}
```

> Note: You may want to create a run script `"schematic:getting-started-with-debugging": "schematics .:tree-debug --project=web-app-with-options --name=DEBUG",` for the specified schematic.

> (OPTIONAL) Update/add `--dry-run=false` to allow the schematic to execute with a live update if all things are good. Make sure your schematic will not have any detrimental side-effects - the default mode is set to `true` for safety.

- [ ] Open the `factory` method of the schematic. Typically, this is the `index.ts` file. Add a break point within the function code.

3-Ways to Debug

1. Hit `F5` and the debugger should launch and break on the break point in the factory method.
2. terminal: `schematics .:tree-debug --project=web-app-with-debugging --name=DEBUG`
3. `npm run schematic:getting-started-with-debugging`

Run the Schematic LIVE

Since, we are in a workspace, we can also run the schematic live against other projects in our workspace. Using the `--project` option allows you to target a specific project. Create a new `application` project to test the schematic.

```
ng g application `web-app-with-options` --style=scss --routing --dry-run
```

If you choose to run the schematic `live`, use `schematics .:tree-debug --project=web-app-with-debugging --name=DEBUG --dry-run=false`. If will output the following in the terminal:

```
 schematics .:tree-debug --project=web-app-with-options --name=DEBUG --dry-run=false
    Preparing to retrieve the project using: web-app-with-options
    Preparing to determine the target path.
    The target path: /apps/web-app-with-options/src/app
    Finished options setup.
CREATE /apps/web-app-with-options/src/app/debug.txt (68 bytes)
```

You should also have a new `debug.txt` file in the specified project - with content:

```
Hello DEBUG, I'm a schematic at Fri, 23 Nov 2018 22:29:51 GMT.
```

If you attempt to run the schematic again, it will not succeed because the `debug.txt` file already exists.

```
ERROR! /apps/web-app-with-options/src/app/debug.txt already exists.
The Schematic workflow failed. See above.
```

However, for the brave and those with specific use cases, you can include the `--force` option and the schematic will ignore the rule that the file already exists.

```
schematics .:tree-debug --project=web-app-with-options --name=DEBUG --dry-run=false --force
    Preparing to retrieve the project using: web-app-with-options
    Preparing to determine the target path.
    The target path: /apps/web-app-with-options/src/app
    Finished options setup.
CREATE /apps/web-app-with-options/src/app/debug.txt (68 bytes)
```

Wrapping Up

There is a lot more you can do with this environment. Kevin Schuchard is also doing a lot with Schematics - he has published several schematics and is currently experimenting with Building Schematics with a Sandbox. Having a sandbox environment that takes advantage of Git (visual diffs) is very nice. Also, it is good to have a safe developer workspace environment where you can develop, test, and apply your schematics without any adverse side-effects to your projects.