@angular/cli

25th July 2018

Khusro Habib

Development Manager/Solution Architect - Engineering Office Dubai

TOGAF, SAP Certified

t: @Khusro_Habib

e: Khusro.Habib@gmail.com

Objective

Provide a jump start on @angular/cli and run through some key commands for developing an angular app

Before we Start

Lets discuss some key facts of angular

Some Key Facts

- Angular is a google framework to develop client side applications
- Developed in type script
- Type script is a super set of javascript
- Angular apps are coded in type scripts which are then compiled into javascript
- Typescript allows the developer to leverage on ECMAScript 6/ES2015 features like classes, interfaces, modules etc.

Modules

NgModules

- Helps in organizing code
- Each angular app has at least one module, called the app module
- An angular app can be organized into features using modules
- A module groups components, provides services, and define routes
- A module can import or export other modules
- Used to group components shared across other modules

Components

Components

- Is defined by adding a @Component decorator to a class
- Defines a class with data and logic
- Linked to an HTML view

```
import { Component } from '@angular/core';

@Component({
    selector: 'abc-employee',
    templateUrl: './employee.component.html',
    styleUrls: ['./employee.component.css']
})
export class EmployeeComponent {
}
```

Directives

Directive

- Is defined by adding a @Directive decorator to a class
- Used to change the appearance or behavior of a DOM element
- Structural Directives e.g. NgFor, NgIf

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
    constructor(el: ElementRef) {
       el.nativeElement.style.backgroundColor = 'yellow';
    }
}
```

```
<p appHighlight> Highlight me! </p>
```

Pipes

Pipes

- Is defined by adding a @Pipe decorator to a class

- Used to format data before displaying, for e.g. currency, lowercase, uppercase

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
        name: 'lowercase'
})
export class LowercasePipe implements PipeTransform {

    transform(value: string, args?: any): any {
        return value.toLocaleLowerCase();
    }
}
```

```
{{ employee.Name | lowercase }}
```

Services

Service

- Can be used to fetch data from the server, log activities etc.

- Inserting the @Injectable decorator will enable dependency injection

```
import { Injectable } from '@angular/core';

@Injectable({
        providedIn: 'root'
})
export class EmployeeDataService {
        constructor() { }
}
```

Starting with angular/cli

@angular/cli

- A command line interface for the angular framework

- https://cli.angular.io

- Command prefix ng
- ng new – to create a new project
- ng generate – to generate components, services, pipes, directives etc.
- ng serve – to test the project

Install the cli

Windows

```
npm install -g @angular/cli
```

Mac OS

```
sudo npm install -g @angular/cli
```

To check the version of cli

```
ng -v
```

Create a New App

- ng new app1 — Create and app and restore npm packages

OPTIONS

-d	dry run
--skip-install	creates a project and skips npm package restore
--prefix <name>	creates a project and changes the default
--routing	creates the route module
--style scss	creates project with

Create a Module

ng generate module login	Create a new module login
OPTIONS	
--routing	enables routing
ng generate module employee	Create a new module employee

Generate Component

ng generate component login	Generate component with the name login

OPTIONS

--flat	don’t create separate folder for the component
-is	in-line styles (doesn’t create a separate CSS)
-it	in-line template (doesn’t create a separate HTML)
-m <module_name>	adds to the mentioned module
--prefix <name>	create with a different prefix from the app

Generate Component Cont..

Short form to generate a component with the name login

ng g c login -d

Generate component with the name login and add to employee module

ng g c login –m employee

Generate component with a name different from the modules will update app module

ng g c customer

Generate Service

ng generate service employee/employee

OR

ng g s employee/employee

Generate a service, needs to be added manual to the module

Generate Directive

ng generate directive shared/custom-directive

OR

ng g d shared/custom-directive

Generate a directive and update the module of the respective folder

ng g d shared/custom-directive –m shared

Generate a directive and update the shared module

Generate Pipes & Guards

ng g p shared/transform

Generate a pipe file, used for transformation of content

ng g p shared/transform –m shared

Generate a pipe file and add to shared module

ng g guard auth

Generate auth guard, these are used to protect application routes

Build and Run App

ng build

By default generates a dev build

--dev	To generate a dev build (Default Option)
--prod	To generate a production build

ng serve

-o	host the app in local server an open in browser

Thank you

Q & A?