How to Install angular Cli?

Please prefer below url to install Angluar Cli.

https://cli.angular.io/

How to create New project Using Angular CLI ?

To create a project please open Terminal/Command line and run below command

```
ng new App-name
```

What about SPA(Single Page Application)?

A single-page application is an app that works inside a browser and does not require page reloading during use. You are using this type of applications every day. It is just one web page that you visit which then loads all other content using JavaScript—which they heavily depend on.

What is routing?

Routing basically means navigating between pages. You have seen many sites with links that direct you to a new page. This can be achieved using routing. Here the pages that we are referring to will be in the form of components.

The Angular 7 router is an essential element of the Angular platform. It allows developers to build **Single Page Applications** with multiple states and views using routes and components and allows client-side navigation and routing between the various components. It's built and maintained by the core team behind Angular development and it's contained in the `@angular/router` package.

You can use browser url's to navigate between different components.

Angular 7 provides a powerful router that allows you to map browser routes to components. So let's see how we can add routing to applications built using Angular 7.

Here we came to know about various concepts related to Angular routing such as:-

- Components ,routes and paths,
- router outlet
- Route parameters
- Query parameters

Adding <base href>

Initially we need to add <base href> into Index.html File. Please see below code

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>RoutingDemo</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>

<body>
  <app-root></app-root>
</body>

</html>
```

Now lets Create Routing Module.

You would need to create routing module inside main application module . we can create using below command.

```
ng generate module app-routing –module app  --flat
```

The `--flat` option tells the CLI to generate a flat file without a subfolder. This way the `app-routing.module.ts` file will be created in the `src/app` folder along with the `app.module.ts` file. This is the content of this module before setting up routing:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
export class AppRoutingModule { }
```

Importing the Router and Setting up Routing

Now we need to update the routing module as follow:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
const routes: Routes = [];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Now create following components

1) Deppartment List
   1.1. Department Overview
   1.2. Department Contact

2) Empolyee list
3) Page not found component

Now defining the components into app routing module.

Now we need to import the components into **app-routing. Modules.ts** file

```
import { DepartmentListComponent } from './department-list/department-list.component';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { DepartmentOverviewComponent } from './department-overview/department-overview.component';
import { DepartmentContactComponent } from './department-contact/department-contact.component';
```

then we need to define routing components array like below..

```
export const routingComponents = [DepartmentListComponent,
                                  DepartmentDetailComponent,
                                  EmployeeListComponent,
                                  PageNotFoundComponent,
                                  DepartmentOverviewComponent,
                                  DepartmentContactComponent]
```

Then we have to import routingComponents into **app.module.ts** file also pass it to declaration array.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule, routingComponents } from './app-routing.module';

import { AppComponent } from './app.component';


@NgModule({
  declarations: [
    AppComponent,
    routingComponents
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Then we have to define routes array into **app-routing.module.ts** file

So finally our app routing file will look like this.

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { DepartmentListComponent } from './department-list/department-list.component';
import { DepartmentDetailComponent } from './department-detail/department-detail.component';
import { EmployeeListComponent } from './employee-list/employee-list.component';
import { PageNotFoundComponent } from './page-not-found/page-not-found.component';
import { DepartmentOverviewComponent } from './department-overview/department-overview.component';
import { DepartmentContactComponent } from './department-contact/department-contact.component';


const routes: Routes = [
  { path: '', redirectTo: '/departments', pathMatch: 'full' },
  { path: 'departments', component: DepartmentListComponent },
  {
    path: 'departments/:id',
    component: DepartmentDetailComponent,
    children: [
      { path: 'overview', component: DepartmentOverviewComponent},
      { path: 'contact', component: DepartmentContactComponent}
    ]
   },
  { path: 'employees',   component: EmployeeListComponent },
  { path: '**',   component: PageNotFoundComponent }

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingComponents = [DepartmentListComponent,
                                  DepartmentDetailComponent,
                                  EmployeeListComponent,
                                  PageNotFoundComponent,
                                  DepartmentOverviewComponent,
                                  DepartmentContactComponent]
```

In this above file we have defined the we have added object to the routes array.

```
{ path: 'departments', component: DepartmentListComponent },
```

Here url will be **‘/departments’** and it will use **DepartmentListComponent**

You can create a nested routing by defining child routes using the children property of a route (alongside a path and component properties). You also need to add a nested router-outlet in the HTML template related to the component linked to the parent route

```
{
   path: 'departments/:id',
   component: DepartmentDetailComponent,
   children: [
     { path: 'overview', component: DepartmentOverviewComponent},
     { path: 'contact', component: DepartmentContactComponent}
   ]
  },
```

Next add a nested router outlet into department details components.

```
 <router-outlet></router-outlet>
```

To redirect to child routes we have created to functions as follow

```
 showOverview(){
   this.router.navigate(['overview'], { relativeTo: this.route });
 }

 showContact(){
   this.router.navigate(['contact'], { relativeTo: this.route });
 }
```

1. **showoverview()** will redirect to **department-overview** component
2. **showContact()** will redirect to **department-contact** components

```
    <button (click)="showOverview()">Overview</button>
    <button (click)="showContact()">Contact</button>
```

By this button click we can call this function

How to define default route ?

We can define default route like this

```
 { path: '', redirectTo: '/departments', pathMatch: 'full' },
```

if our url http://localhost:4200/ means no value passed to path it will redirect to departments path or departments component.

How to redirect to 404 page ?

If path does not match then we have can redirect to 404 page but we need to define this as the last element of **Routes** Array

```
{ path: '**',   component: PageNotFoundComponent }
```