Shubham Rout · Follow
5 min read · May 20, 2024

MICROFRONTEND APPLICATION USING ANGULAR 17 (STANDALONE).

Single SPA is a JavaScript Library that helps create micro frontend(MF) applications. (node.js v18. 13 or newer)

How to create an angular micro frontends using single SPA?

We need to globally install the single SPA CLI before beginning —

```
npm i -g create-single-spa
```

After successful installation we are ready to create our first micro frontend. We can create MFs that can use any of the JavaScript libraries. But here we will be creating Angular MFs. In the terminal hit command.

```
create-single-spa
```

It will ask for the following data in order to create the single spa project

? Directory for new project (.) -> testProject

? Select type to generate -> as we are creating the project for the first time , so we need to select the **single-spa root config** option from the list.

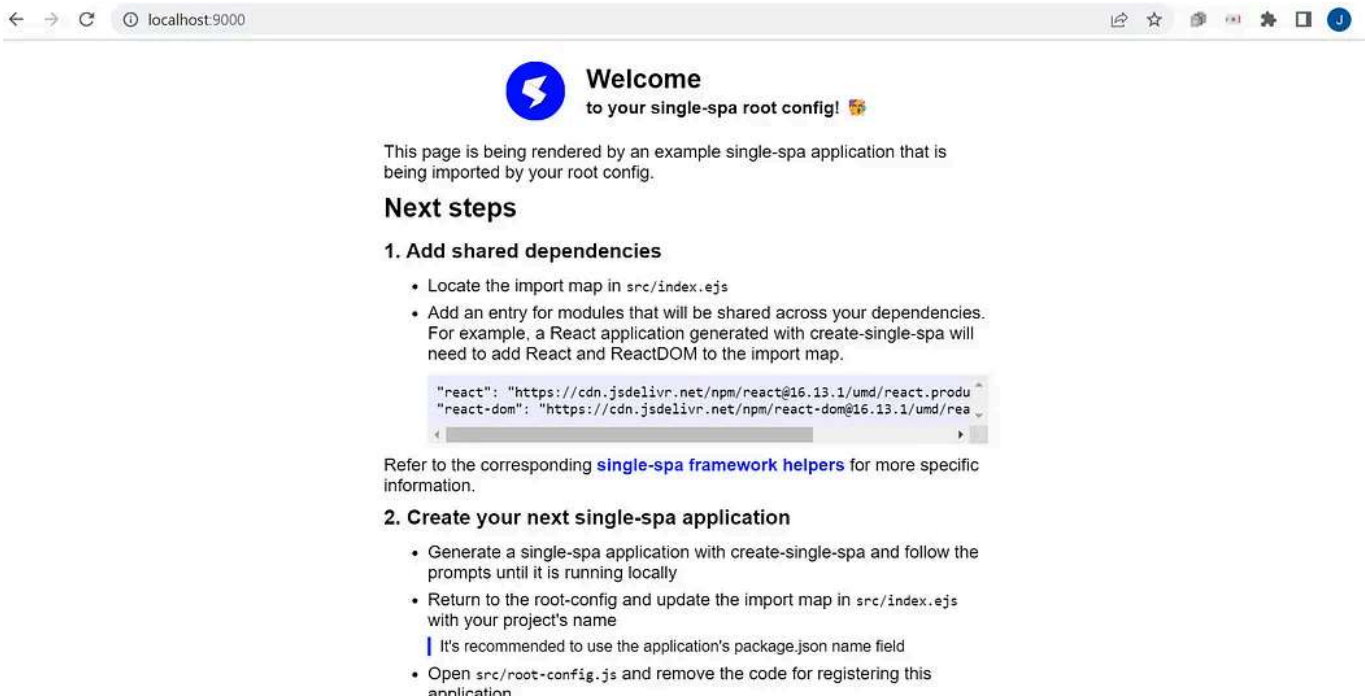
? Which package manager do you want to use? -> as we are crating angular project , and uses the npm packages, so we need to select the **npm** as the manager.

? Will this project use Typescript? (y/N) -> obvious it's yes

? Would you like to use single-spa Layout Engine (Y/n) -> if you want to use the single spa layout engine , select Y else N

? Organization name (can use letters, numbers, dash or underscore) -> enter you organization name here, it can be anything nothing specific. Your root config is getting created, on completion enter **npm install** in the root directory, and enter **npm start** to run the root config.

Your project running information will appear in the terminal screen, in most of the cases it runs in the 9000 port , so please open **http://localhost:9000/** in your browser, and observe the following screen will appear in your window.



Congratulations, your app-root is working fine, the next step is to create the angular MFs.

Navigate into the directory created and follow the following steps. In order to create the angular apps, only few places we need to change options that are asked at the time of root config creations.

In order to create a microfrontend open a new terminal and execute the command given below

```
create-single-spa
```

follow the given steps below

```
? Directory for new project .
? Select type to generate single-spa application / parcel
? Which framework do you want to use? angular
? Project name (can use letters, numbers, dash or underscore) login
? Which stylesheet format would you like to use? Sass (SCSS)     [
https://sass-lang.com/documentation/syntax#scss                ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (
No
```

After the succesfull execution, the boilerplate files will be created. Then

it will ask for enabling angular's analytics.(choose according to your requirement)

```
? Would you like to share pseudonymous usage data about this project with the An
at Google under Google's Privacy Policy at https://policies.google.com/privacy.
```

```
details and how to change this setting, see https://angular.io/analytics. Yes
```

It will then install *single-spa-angular*.

```
The package single-spa-angular@9.1.2 will be installed and executed.
Would you like to proceed? Yes
```

It will then ask for routing and port .

```
? Does your application use Angular routing? Yes
? What port should your project run on? 4200
```

The next steps are to make it functional —

Step -1 → Open src/index.ejs file inside root directory and add the location where the app is hoisted inside the import map. As of now it is our local host, but in future the location would be your deployed server location.

```
<% if (isLocal) { %>
<script type="systemjs-importmap">
  {
    "imports": {
      "@single-spa/welcome": "https://unpkg.com/single-spa-welcome/dist/single-spa-
welcome.js",
      "@isu/root-config": "//localhost:9000/isu-root-config.js",
      "login": "//localhost:4200/main.js"
    }
  }
</script>
<% } %>
```

Step-2 → For running angular app we need to enable the zone script in index.ejs file as shown in the below image

```
<script src="https://cdn.jsdelivr.net/npm/zone.js@0.11.3/dist/zone.min.js"></script>
```

Step -3 → Open microfrontend-layout.html and change the followings ->

inside route tag add the route path to your newly created MF like —

```
<single-spa-router>
  <main>
    <route default>
      <application name="@single-spa/welcome"></application>
    </route>
    <route path="login">
      <application name="login"></application>
    </route>
  </main>
</single-spa-router>
```

Step -4 → Then do the following changes in angular.json inside login folder

angular.json U ×

login > angular.json > {} projects > {} login > {} architect > {} build > {} options > [ ] assets > 0

```
5     "projects": {
6       "login": {
15        "prefix": "app",
16        "architect": {
17          "build": {
18            "builder": "@angular-builders/custom-webpack:browser",
19            "options": {
20              "outputPath": "dist/login",
21              "index": "src/index.html",
22              "browser": "src/main.ts",
23              "polyfills": [
24                "zone.js"
25              ],
26              "tsConfig": "tsconfig.app.json",
27              "inlineStyleLanguage": "scss",
28              "assets": [
29                "src/favicon.ico",
30                "src/assets"
31              ],
32              "styles": [
33                "src/styles.scss"
34              ],
35              "scripts": [],
36              "main": "src/main.single-spa.ts",
37              "customWebpackConfig": {
38                "path": "extra-webpack.config.js",
39                "libraryName": "login",
```

Delete this

Move to

Step -5→Then navigate to login/src/main.single-spa.ts and we can see that the lifecycle has been configured with AppModule but in angular 17 we no longer use AppModule .

```
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

const lifecycles = singleSpaAngular({
  bootstrapFunction: (singleSpaProps) => {
    singleSpaPropsSubject.next(singleSpaProps);
    return platformBrowserDynamic(getSingleSpaExtraProviders()).bootstrapModule(
      AppModule
    );
  },
  template: '<app-root />',
  Router,
  NavigationStart,
  NgZone,
});
```

So we need to do the neccessary changes below

- Remove the environment file for now and the condition for enabling production.(if we need environment for production development and staging create an environment file in login/src/environment and do the environment setup)
- instead of AppModule we will use AppComponent along with some providers configurations.

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app/app.component';
import { APP_BASE_HREF } from '@angular/common';
import { EmptyRouteComponent } from './app/empty-route/empty-route.component';

const lifecycles = singleSpaAngular({
  bootstrapFunction: (singleSpaProps) => {
    singleSpaPropsSubject.next(singleSpaProps);
    return bootstrapApplication(AppComponent, {
      providers: [
        { provide: APP_BASE_HREF, useValue: '/' },
        getSingleSpaExtraProviders(),
        provideRouter([{ path: '**', component: EmptyRouteComponent }]),
      ],
    });
  },
  template: '<app-root />',
  Router,
  NavigationStart,
  NgZone,
});
```

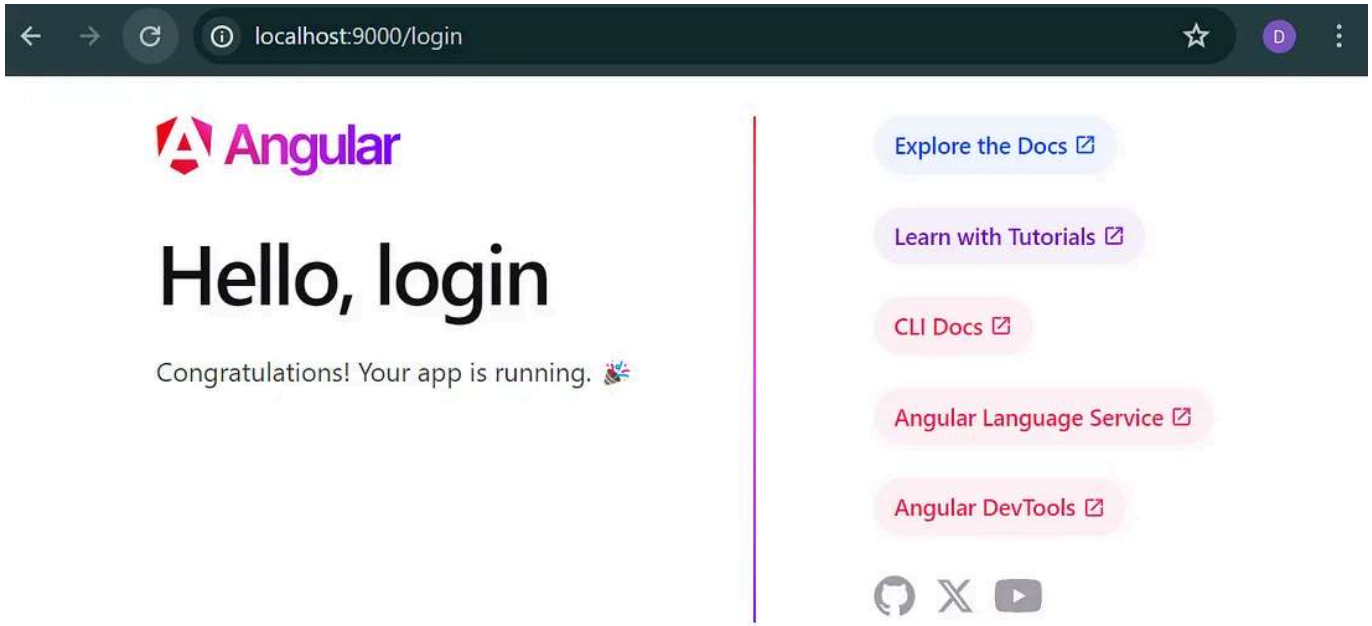
The final step -

Now enter command in the newly created app terminal —

```
npm run serve:single-spa:login
```

If it returns any error, then hit **npm i**, then run the above command

And open you browser now and change the URL path as mentioned in **microfrontend-layout.html.** (in our case open **http://localhost:9000** for root file and **http://localhost:9000/login** for login).



Congratulations again. You have successfully created MF and its working too in angular 17 (standalone application).

Micro Front End | Angular | Angular 17

Written by Shubham Rout

10 Followers · 7 Following

Software Developer

Follow

Responses (2)

What are your thoughts?

Respond

subhakanta tarai
7 months ago

Great,it helped me a lot.....

1 1 reply Reply

Thiago B A Salles
about 2 months ago

How can I work with subpaths in standalone components? With my current configuration, the path /login works with app.component.ts. However, when I create a test.component (with HTML, CSS, and JS in the path /test) and register a route in.....

Read More

Reply

Recommended from Medium

In ITNEXT by Maksim Dolgikh

Angular 19. Trying to stay afloat

The story of how, in the pursuit of all things Angular has already stopped realizing what i...

Dec 8 1.1K 11

In Angular Blog by Minko Gechev

Meet Angular v19

In the past two years we doubled down on our investment in developer experience and...

Nov 19 3K 52

Lists

General Coding Knowledge

20 stories · 1825 saves

Priyabrata Saha

Mastering Angular: Best Practices to Code Like a Pro in 2025

As Angular continues evolving, coding practices must adapt to align with its latest...

Dec 10 89 3

In Stackademic by Yue Wang

Deploying Angular SSR （v17, 18 & 19） Websites on AWS Lambda

Since version 17, Angular has made significant improvements to SSR (Server-Si...

Nov 16 58 1

In Coding Beauty by Tari Ibaba

This new JavaScript operator is an absolute game changer

Say goodbye to try-catch

Volodymyr Golosay

Don’t Overcomplicate Your Angular App

Sep 18 6K 89

Avoiding Common Overengineering Pitfalls in Angular Apps

Sep 12 481 5

See more recommendations