angular micro frontend

install

npm i create-nx-workspace

run commands

npx create-nx-workspace ng-mf

Select "Integrated Monorepo" then "Angular"

npx nx g @nrwl/angular:host dashboard

npx nx g @nrwl/angular:remote login --host=dashboard

Adding Functionality

nx g @nrwl/angular:lib shared/data-access-user

if has error use npx nx ....

nx g @nrwl/angular:service user --project=shared-data-access-user

if has error use npx nx ....

This will create a file user.service.ts under the shared/data-access-user library. Change its contents to match:

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

import { BehaviorSubject } from 'rxjs';

@Injectable({

providedIn: 'root',

})

export class UserService {

private isUserLoggedIn = new BehaviorSubject(false);

isUserLoggedIn$ = this.isUserLoggedIn.asObservable();

checkCredentials(username: string, password: string) {

if (username === 'demo' && password === 'demo') {

this.isUserLoggedIn.next(true);

}

}

logout() {

this.isUserLoggedIn.next(false);

}

}  
  
Add a new export to the shared/data-access-user's index.ts file:

export \* from './lib/user.service';

**Login Application**

First, add FormsModule to the imports array in your remote-entry/entry.module.ts file

import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

import { RouterModule } from '@angular/router';

import { RemoteEntryComponent } from './entry.component';

import { NxWelcomeComponent } from './nx-welcome.component';

import { remoteRoutes } from './entry.routes';

import { FormsModule } from '@angular/forms';

@NgModule({

declarations: [RemoteEntryComponent, NxWelcomeComponent],

imports: [CommonModule, FormsModule, RouterModule.forChild(remoteRoutes)],

providers: [],

})

export class RemoteEntryModule {}

Next we want to set up our entry.component.ts file so that it renders a login and has injected our UserService to allow us to sign the user in:

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

import { UserService } from '@ng-mf/shared/data-access-user';

@Component({

selector: 'ng-mf-login-entry',

template: `

<div class="login-app">

<form class="login-form" (ngSubmit)="login()">

<label>

Username:

<input type="text" name="username" [(ngModel)]="username" />

</label>

<label>

Password:

<input type="password" name="password" [(ngModel)]="password" />

</label>

<button type="submit">Login</button>

</form>

<div \*ngIf="isLoggedIn$ | async">User is logged in!</div>

</div>

`,

styles: [

`

.login-app {

width: 30vw;

border: 2px dashed black;

padding: 8px;

margin: 0 auto;

}

.login-form {

display: flex;

align-items: center;

flex-direction: column;

margin: 0 auto;

padding: 8px;

}

label {

display: block;

}

`,

],

})

export class RemoteEntryComponent {

username = '';

password = '';

isLoggedIn$ = this.userService.isUserLoggedIn$;

constructor(private userService: UserService) {}

login() {

this.userService.checkCredentials(this.username, this.password);

}

}

Let's add a route to our Login application so that we can render the RemoteEntryComponent.

Open app.module.ts and add the following route to the RouterMoodule.forRoot(...) declaration.

RouterModule.forRoot(

[

{

path: '',

loadChildren: () =>

import('./remote-entry/entry.module').then((m) => m.RemoteEntryModule),

},

],

{ initialNavigation: 'enabledBlocking' }

);

next

nx run login:serve

We can see if we navigate a browser to http://localhost:4201 that we see the login form rendered.

If we type in the correct username and password (demo, demo), then we can also see the user gets authenticated!

Perfect! Our login application is complete.

**Dashboard Application**

app.component.ts

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { distinctUntilChanged } from 'rxjs/operators';

import { UserService } from '@ng-mf/shared/data-access-user';

@Component({

selector: 'ng-mf-root',

template: `

<div class="dashboard-nav">Admin Dashboard</div>

<div \*ngIf="isLoggedIn$ | async; else signIn">

You are authenticated so you can see this content.

</div>

<ng-template #signIn><router-outlet></router-outlet></ng-template>

`,

})

export class AppComponent implements OnInit {

isLoggedIn$ = this.userService.isUserLoggedIn$;

constructor(private userService: UserService, private router: Router) {}

ngOnInit() {

this.isLoggedIn$

.pipe(distinctUntilChanged())

.subscribe(async (loggedIn) => {

// Queue the navigation after initialNavigation blocking is completed

setTimeout(() => {

if (!loggedIn) {

this.router.navigateByUrl('login');

} else {

this.router.navigateByUrl('');

}

});

});

}

}

**nx serve dashboard --devRemotes=login**

**replace Static Module Federation with Dynamic Module Federation.**

**Add a module-federation.manifest.json file to the src/assets/ folder in our Dashboard application with the following content:**

{

"login": "http://localhost:4201"

}

Change:

*src/main.ts*

import { setRemoteDefinitions } from '@nrwl/angular/mf';

fetch('/assets/module-federation.manifest.json')

.then((res) => res.json())

.then((definitions) => setRemoteDefinitions(definitions))

.then(() => import('./bootstrap').catch((err) => console.error(err)));

**module-federation.config.js in apps/dashboard**

module.exports = {

name: 'dashboard',

remotes: [],

};

**app.module.ts in src/app change route:**

{

path: 'login',

loadChildren: () =>

import('login/Module').then((m) => m.RemoteEntryModule),

}  
to 🡺

{

path: 'login',

loadChildren: () =>

loadRemoteModule('login', './Module').then(

(m) => m.RemoteEntryModule

),

}

nx serve dashboard --devRemotes=login

**Run nx graph to see a diagram of the dependencies of the projects.**

**Run npx nx connect-to-nx-cloud to enable remote caching and make CI faster.**

**Visit the [Nx Documentation](https://nx.dev/) to learn more.**

**Creating a new Dynamic Host application**

**Generate the Employee Host application**

**nx g @nrwl/angular:host employee --remotes=login –dynamic**

**Generate the Todo Remote application**

**nx g @nrwl/angular:remote todo --host=employee**