# Create Parent Application using Single-SPA

The Parent Application is created as Host for the child apps using Single-SPA

**Steps to Create:**

1. Create new Angular Application
    a. **ng new <PROJECT_NAME> --routing --prefix <PREFIX>**(**Note**: Adding prefix is recommended, by default angular adds it as app-root and when multiple application used the same name the parent application cannot identify the child.
    b. Choose the styling for your project
    c. **cd <PROJECT_NAME>**
2. Install Single-SPA related packages
    a. ng add single-spa-angular
        i. Choose **Routing** '**Yes**' if your project has routing involved else 'No'
        ii. **BrowserAnimationModule** '**Yes**'

        Ensure that the below files are created under your project folder

        **src/main.single-spa.ts**

        **src/single-spa/asset-url.ts**

        **src/single-spa/single-spa-props.ts**

        **package.json** with two new commands under scripts(to build and run) Remove them
3. Explicitly install the dependencies below
    a. **npm i single-spa**
    b. **npm i @angular-builders/custom-webpack**
    c. **npm i import-map-overrides**
    d. **npm i systemjs**
    e. **npm i @types/systemjs**

---

Go to **angular.json**

Replace “**main.single-spa.ts**” “**main.ts**”

Inside **projects build scripts**

**angular.json -> scripts**

```
"scripts": [
        "node_modules/systemjs/dist/system.min.js",
    "node_modules/systemjs/dist/extras/amd.min.js",
    "node_modules/systemjs/dist/extras/named-exports.min.js",
    "node_modules/systemjs/dist/extras/named-register.min.js",
    "node_modules/import-map-overrides/dist/import-map-overrides.js"
]
```

Go to **main.ts** and replace content as below

**main.ts**

```
import { enableProdMode, NgZone } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { PlatformLocation } from '@angular/common';

import { start as singleSpaStart } from 'single-spa';
import { getSingleSpaExtraProviders } from 'single-spa-angular';

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

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

singleSpaStart();

const appId = 'container-app';

platformBrowserDynamic(getSingleSpaExtraProviders()).bootstrapModule(AppModule)
.then(module => {
        NgZone.isInAngularZone = () => {
                // @ts-ignore
                return window.Zone.current._properties[appId] === true;
        };

        const rootPlatformLocation = module.injector.get(PlatformLocation) as any;
        const rootZone = module.injector.get(NgZone);

        // tslint:disable-next-line:no-string-literal
        rootZone['_inner']._properties[appId] = true;
        rootPlatformLocation.setNgZone(rootZone);
})
.catch(err => console.error(err));
```

Go to **extra-webpack.config.js** and replace the content with below

**extra-webpack.config.js**

```
module.exports = (angularWebpackConfig, options) => {
  return {
    ...angularWebpackConfig,
    module: {
      ...angularWebpackConfig.module,
      rules: [
        ...angularWebpackConfig.module.rules,
        {
          parser: {
            system: false
          }
        }
      ]
    }
  };
}
```

**tsconfig.app.json -> compilerOptions**

```
"compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": ["systemjs"]
  },
  "files": [
    "src/main.ts",
    "src/polyfills.ts"
  ]
```

Go to index.html and add the below line inside **<head>** tag

**index.html -> head**

```
<scripttype="systemjs-importmap"src="/assets/import-map.json"></script>
```

Add new file under assets folder

**assets  import-map.json**

Sample code on importing child below

**import-map.json**

```
{
"imports":{
"child1":"http://localhost:4207/main.js",
"child2":"http://localhost:4202/main.js",
"child3":"http://localhost:4208/main.js"
}
}
```

**app.module.ts -> providers**

```
providers:[{
        provide:RouteReuseStrategy,
        useClass:MicroFrontendRouteReuseStrategy
}]
```

Create a new folder “services” under “src” folder and add 3 new files with content as below

**src  services**

**route-reuse-strategy.ts**

```
import { RouteReuseStrategy, ActivatedRouteSnapshot, DetachedRouteHandle } from '@angular/router';
import { Injectable } from '@angular/core';

@Injectable()
export class MicroFrontendRouteReuseStrategy extends RouteReuseStrategy {
  shouldDetach(): boolean {
    return false;
  }

  store(): void { }

  shouldAttach(): boolean {
    return false;
  }

  retrieve(): DetachedRouteHandle {
    return null;
  }

  shouldReuseRoute(future: ActivatedRouteSnapshot, curr: ActivatedRouteSnapshot): boolean {
    /// If a child app routes inside of itself, this app will interpret that as a route change.
    ///
    /// By default, this will result in the current component being destroyed and replaced with a new instance
    /// of the same spa-host component.
    ///
    /// This route reuse strategy looks at the routeData.app to determine if the new route should be
    /// treated as the exact same route as the previous, ensuring we don't remount a child app when said child
app
    /// routes inside of itself.

    return future.routeConfig === curr.routeConfig || (future.data.app && (future.data.app === curr.data.app));
  }
}
```

**single-spa.service.ts**

```
import { Injectable } from '@angular/core';
import { mountRootParcel, Parcel, ParcelConfig } from 'single-spa';
import { Observable, from } from 'rxjs';
import { tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root',
})
export class SingleSpaService {
  private loadedParcels: {
    [appName: string]: Parcel;
  } = {};

  mount(appName: string, domElement: HTMLElement): Observable<unknown> {
    return from(System.import<ParcelConfig>(appName)).pipe(
      tap((app: ParcelConfig) => {
        this.loadedParcels[appName] = mountRootParcel(app, {
          domElement
        });
      })
    );
  }

  unmount(appName: string): Observable<unknown> {
    return from(this.loadedParcels[appName].unmount()).pipe(
      tap(() => delete this.loadedParcels[appName])
    );
  }
}
```

**single-spa.service.spec.ts**

```
import { TestBed } from '@angular/core/testing';

import { SingleSpaService } from './single-spa.service';

describe('SingleSpaService', () => {
  beforeEach(() => TestBed.configureTestingModule({}));

  it('should be created', () => {
    const service: SingleSpaService = TestBed.get(SingleSpaService);
    expect(service).toBeTruthy();
  });
});
```

---

To create host module to integrate the child apps follow the below steps

Create folder spa-host under app (**app  spa-host**)

Add 3 new files with content as below

**spa-host.component.ts**

```
import { Component, OnInit, ViewChild, ElementRef, ChangeDetectionStrategy } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Observable } from 'rxjs';

import { SingleSpaService } from '../../services/single-spa.service';

@Component({
  selector: 'app-spa-host',
  template: '<div #appContainer></div>',
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class SpaHostComponent implements OnInit {

  constructor(private singleSpaService: SingleSpaService, private route: ActivatedRoute) { }

  @ViewChild('appContainer', { static: true })
  appContainerRef: ElementRef;

  appName: string;

  mount(): Observable<unknown> {
    return this.singleSpaService.mount(this.appName, this.appContainerRef.nativeElement);
  }

  unmount(): Observable<unknown> {
    return this.singleSpaService.unmount(this.appName);
  }

  ngOnInit() {
    this.appName = this.route.snapshot.data.app;
    this.mount().subscribe();
  }
}
```

**spa-host.module.ts**

```
import { RouterModule, Routes } from '@angular/router';
import { CommonModule } from '@angular/common';
import { NgModule } from '@angular/core';

import { SpaUnmountGuard } from './spa-unmount.guard';
import { SpaHostComponent } from './spa-host.component';

const routes: Routes = [
  {
    path: '',
    canDeactivate: [SpaUnmountGuard],
    component: SpaHostComponent,
  },
];

@NgModule({
  declarations: [SpaHostComponent],
  imports: [CommonModule, RouterModule.forChild(routes)]
})
export class SpaHostModule { }
```

**spa-unmount.guard.ts**

```
import { Injectable } from '@angular/core';
import { CanDeactivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
import { Observable } from 'rxjs';
import { map } from 'rxjs/operators';

import { SpaHostComponent } from './spa-host.component';

@Injectable({ providedIn: 'root' })
export class SpaUnmountGuard implements CanDeactivate<SpaHostComponent> {
  canDeactivate(
    component: SpaHostComponent,
    currentRoute: ActivatedRouteSnapshot,
    currentState: RouterStateSnapshot,
    nextState: RouterStateSnapshot
  ): boolean | Observable<boolean> {
    const currentApp = component.appName;
    const nextApp = this.extractAppDataFromRouteTree(nextState.root);

    if (currentApp === nextApp) {
      return true;
    }

    return component.unmount().pipe(map(_ => true));
  }

  extractAppDataFromRouteTree(routeFragment: ActivatedRouteSnapshot): string {
    if (routeFragment.data && routeFragment.data.app) {
      return routeFragment.data.app;
    }

    if (!routeFragment.children.length) {
      return null;
    }

    return routeFragment.children.map(r => this.extractAppDataFromRouteTree(r)).find(r => r !== null);
  }
}
```

---

**Routing:**

Routing should be handled explicitly with the path having the child name and also pass the data property to identify the respective child

**Routing**

```
{
   path:'child1',
 children:[{
        path:'**',
        loadChildren:()=>import('./spa-host/spa-host.module').then(m=>m.SpaHostModule),
data:{app:'child1'}
 }]
},
```