Create Child Application as Webcomponent

The Child Application is created as Web component 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** '**No**'

        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)
3. Explicitly install the dependencies below
    a. **npm i single-spa**
    b. **npm i @angular-builders/custom-webpack**

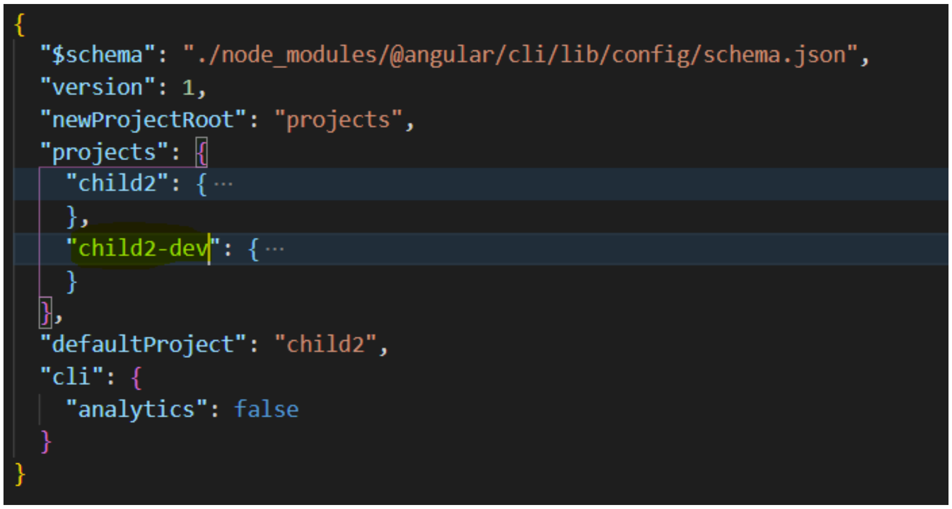
Once this above steps are done the application is ready to be converted into web component

---

For **Local Debugging** of the application follow the below steps

Go to **angular.json**

Inside **project** section, copy the default project and add it back under same project with **-dev** postfix



```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "child2": {···
    },
    "child2-dev": {···
    }
  },
  "defaultProject": "child2",
  "cli": {
    "analytics": false
  }
}
```

In the above example of “child2-dev”

Modify the below content inside

Replace:

“child2”   “child2-dev”

“@angular-builders/custom-webpack”   “@angular-devkit/build-angular”

“main.single-spa.ts” “main.ts”

“tsconfig.app.json” “tsconfig-dev.app.json”

Delete:

“~~customWebPackConfig~~” section

Go to **package.json**

Add a new script under scripts section

"**serve:development**": "ng s --project <CREATED_PROJECT_UNDER_ANGULAR_JSON> --port <PORT_NUMBER>"

Eg:

"**serve:development**": "ng s --project child2-dev --port 4201"

Add a new file **tsconfig-dev.app.json** and add the below content

tsconfig-dev.app.json

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "outDir": "./out-tsc/app",
    "types": []
  },
  "files": [
    "src/main.ts",
    "src/polyfills.ts"
  ],
  "include": [
    "src/**/*.d.ts"
  ]
}
```

To run locally **npm run serve:development**

To build and create package **npm run build:single-spa:<PROJECT_NAME>**

To serve and test in the Parent application **npm run serve:single-spa:<PROJECT_NAME>**

Converting existing Application as Web Component

Root Name Changes:

Modify the name **app-root** into **PROJECT_NAME-app-root** in the below files (To ensure there is no conflict because of the default naming)

index.html

app.component.ts

main.single-spa.ts

Check if any other places have the reference and modify all occurrences

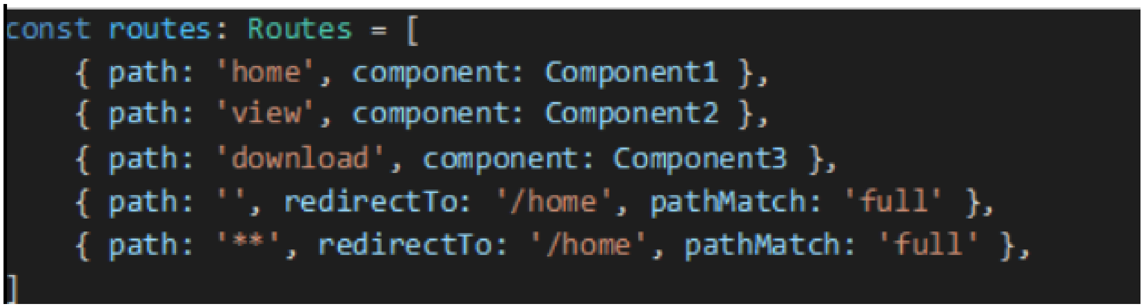
Eg: <app-root></app-root> **<child-app-root></child-app-root>**

Routing Changes:

Since the application will be integrated to the Parent Application the routing should also match the module of the parent routing

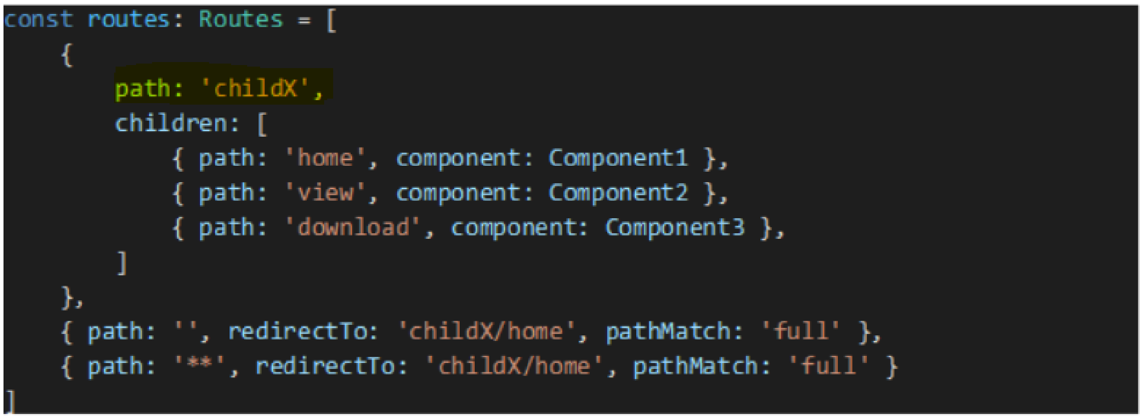
http://<PARENT_APP>/childX/<some_route_inside_childX>

Consider this was the initial state of the routing



```
const routes: Routes = [
    { path: 'home', component: Component1 },
    { path: 'view', component: Component2 },
    { path: 'download', component: Component3 },
    { path: '', redirectTo: '/home', pathMatch: 'full' },
    { path: '**', redirectTo: '/home', pathMatch: 'full' },
]
```

The new route should be as below



```
const routes: Routes = [
    {
        path: 'childX',
        children: [
            { path: 'home', component: Component1 },
            { path: 'view', component: Component2 },
            { path: 'download', component: Component3 },
        ]
    },
    { path: '', redirectTo: 'childX/home', pathMatch: 'full' },
    { path: '**', redirectTo: 'childX/home', pathMatch: 'full' }
]
```

Image Changes:

To include image in the project the approach is different with Single-SPA

Existing:

const imageURL = ‘./assets/image.png’

New:

To Add Image

```
import { assetUrl } from 'src/single-spa/asset-url';

export class SomeComponent {
        // When using Single-SPA
        imageUrl = assetUrl('./assets/image.png');
}
```

For more clarification visit the documentation of Single-SPA

https://single-spa.js.org/docs/ecosystem-angular/#angular-versions

---