BPA使用手冊建立自定義MFE v4.1.2

目錄

自定義MFE建立

MicroFrontEnd(MFE)是一種架構模式，在構建時或運行時會合成使用者介面(UI)頁面的不同片段。每個MFE都是獨立開發的。

建立MFE

使用任何框架開發的應用程式都可以作為MFE進行載入。以下部分是一個將Angular應用程式新增為MFE的示例。

建立角度應用程式

1. 運行以下命令以建立名為test-mfe的Angular應用程式。

```
ng new test-mfe –routing –style=scss
```

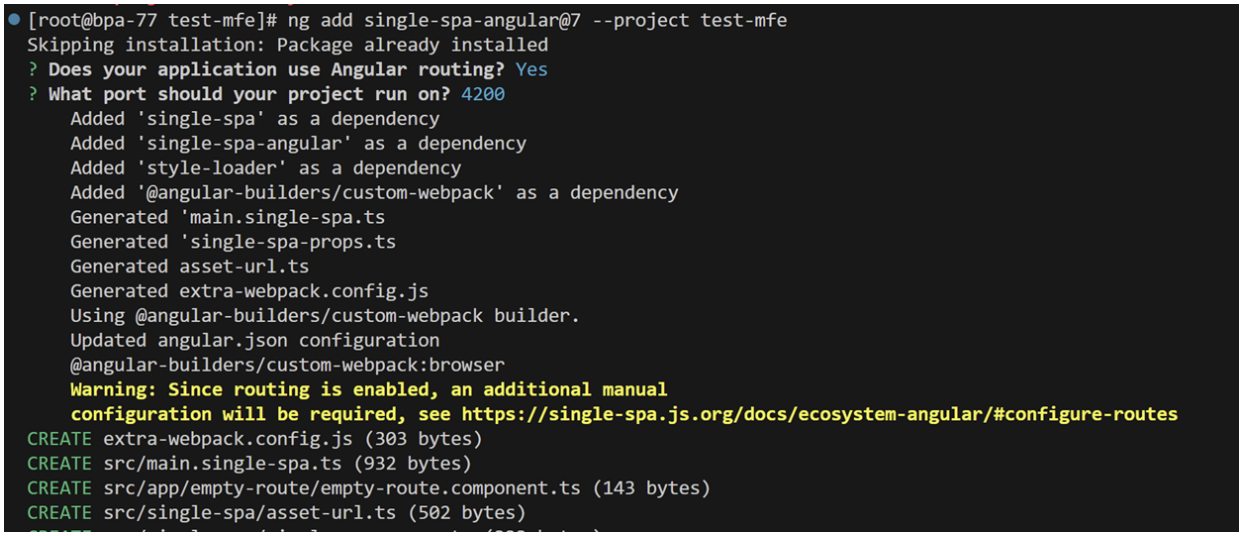
```
[root@bpa-77 MFE]# ng new test-mfe --routing --style=scss
CREATE test-mfe/README.md (1061 bytes)
CREATE test-mfe/.editorconfig (274 bytes)
CREATE test-mfe/.gitignore (548 bytes)
CREATE test-mfe/angular.json (3106 bytes)
CREATE test-mfe/package.json (1039 bytes)
CREATE test-mfe/tsconfig.json (863 bytes)
CREATE test-mfe/.browserslistrc (600 bytes)
CREATE test-mfe/karma.conf.js (1425 bytes)
CREATE test-mfe/tsconfig.app.json (287 bytes)
```

test-mfe

2. 運行cd命令以切換到test-mfe。
3. 運行以下命令在test-mfe中新增單個spa:

```
ng add single-spa-angular@7 –project test-mfe
```

附註：使用了相容版本的Angular 14和single-spa-angular@7。



Test-mfe中的單Spa

4. 在呼叫上一個命令期間指定用於本地開發的埠。在「angular.json」中新增與deployUrl相同的設定。

附註：deployUrl設定僅用於本地開發。

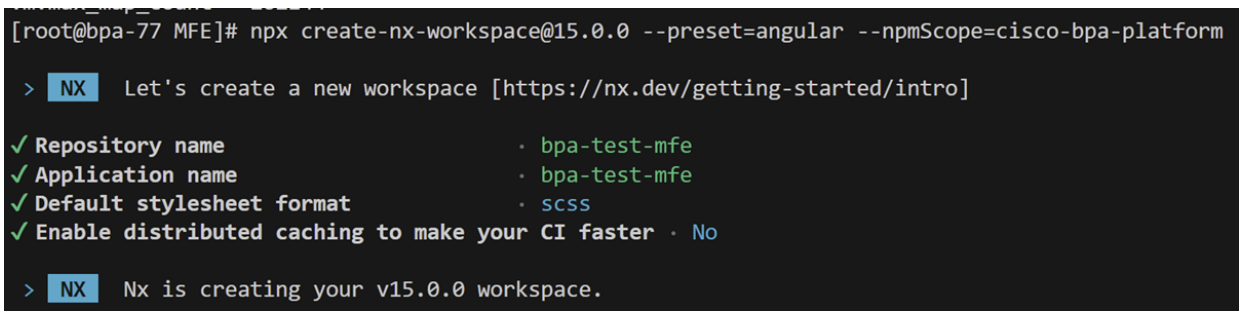
```
"deployUrl": "http://localhost:4201/"
```

在Nx Monorepo中建立角度應用程式

1. 運行以下命令，在Nrwl Extensions(Nx)工作區內建立Angular應用程式：

```
npx create-nx-workspace@15.0.0 –preset=angular –npmScope=cisco-bpa-platform
```

2. 提供使用者輸入，如下例所示：



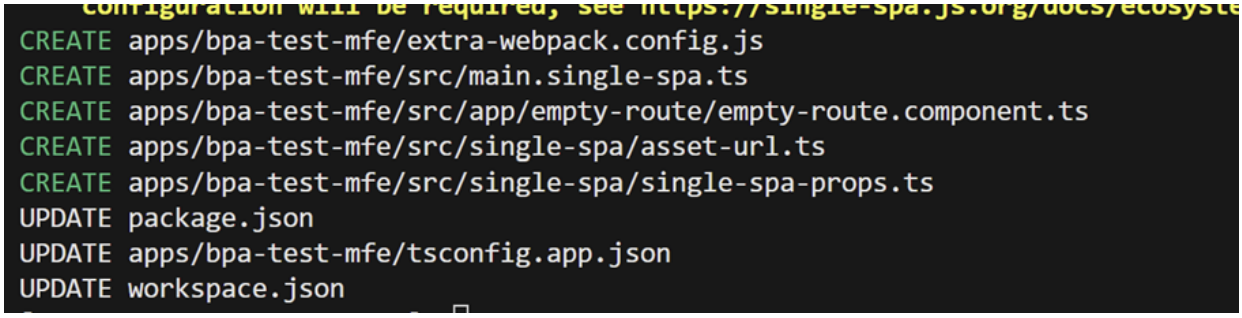
使用者輸入

bpa-test-mfe Nx工作區是使用bpa-test-mfe角度應用程式建立的。

3. 應用「單spa — 角」圖表來配置新建立的角度應用程式。版本根據角度版本而變化。
4. 運行cd命令以切換到test-mfe。
5. 運行以下命令在bpa-test-mfe中新增單個spa:

```
npm install single-spa-angular@7 \--legacy-peer-deps && npx nx g
single-spa-angular:ng-add \--project bpa-test-mfe
```

建立和更新了以下檔案：



bpa-test-mfe中的單一spa

6. 在呼叫上一個命令期間指定用於本地開發的埠。在「project.json」中作為deployUrl新增相同的設定。

附註：deployUrl設定僅用於本地開發。

```
"deployUrl": "http://localhost:4201",
```

Angular和Nx Angular應用的常見更改

檢查Webpack版本

更新與Angular版本對應的「@angular-builders/custom-webpack」版本。以下示例中的版本用於Angular 14應用程式。

```
"@angular-builders/custom-webpack": "14.1.0",
```

更新選擇器

在./src/main.single-spa.ts中使用唯一選擇器「<app-test-mfe-root>」更新模板。

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

更新選擇器

在根元件中更新選擇器

在./src/app/app.component.ts中更新相同的選擇器名稱「<app-test-mfe-root>」。

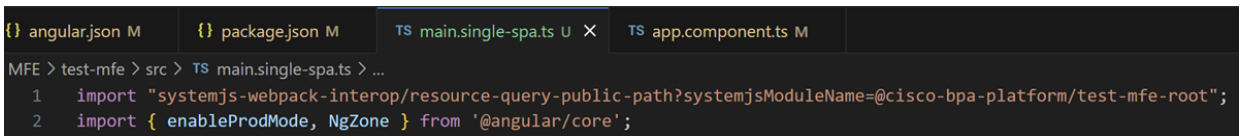
在根元件中更新選擇器

安裝systemsjs-web-interop

「systemsjs-web-interop」 NPM包用於動態設定資源的公共路徑。

```
systemjs-webpack-interop": "^2.3.7",
```

安裝「systemsjs-web-interop」，並將下面的行新增為「main.single-spa.ts」檔案中的第一個匯入語句。



main.single-spa.ts中的行

systemjsModuleName應是唯一的，並且用作標識MFE的名稱，如上所述。上述更改有助於解決從MFE的相應主機緩慢載入的資產問題。

更新路由

在新角度應用程式的「app.routes.ts」檔案中包括應用程式特定的路由。

 附註：必須包括「app.routes.ts」的以下路徑。

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

自註冊MFE應用程式的微服務更改

大多數UI應用程式在設計上都與微服務相關聯。應該執行下面各節中的以下步驟，以便相關聯的微服務將相關聯的UI應用程式載入為MFE。

定義MFE詳細資訊

1. 在microservice/src/config/mfe.json中建立「mfe.json」檔案。

```
[
    {
        "layoutenabled" : true,
        "name" : "@cisco-bpa-platform/test-mfe-root",
        "type": "application",
        "path": "ex/test-mfe",
        "display_name": "Test MFE",
        "menu_path": "Application",
        "custom": true,
        "default": false,
        "distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
        "version": "4.1.0-501"
    }
]
```

2. 新增MFE的詳細資訊，如下所示：

- 自定義:指示此應用程式是否為自定義應用程式
- Layout_enabled:如果需要標頭，則啟用此屬性
- 名稱:MFE應用程式的唯一名稱
- 顯示名稱:側面導航中顯示為選單項的名稱
- Type:指示MFE應用程式
- Menu_path:表示將MFE應用程式作為選單項新增到其中的導航選單；與自定義屬性結合使用
- 分發路徑:指示載入MFE應用程式的靜態檔案

- 路徑：指示MFE應用程式關聯的URL路由
- 預設:如果設定為true，則路徑不應存在；如果設定為false，則路徑存在

註冊MFE

呼叫核心API端點（例如POST mfe/應用）以加入各自的MFE。 為了簡化過程，將建立addMfeApps實用程式方法以接受MFE資訊。註冊BPA "mfe.json"檔案中定義的MFE詳細資訊。

1. 從微服務「app.js」中的公共應用程式匯入mfeHelper。

```
const { mfeHelper } = require('@cisco-bpa-platform/mw-util-common-app');
```

2. 使用addMfeApps()方法註冊MFE詳細資訊。這會在「mongo db core_db -> mfe_apps」集合中新增一個條目。

```
async function mfeAppProcessor() {
  try {
    var data = require('./config/mfe.json');
    await mfeHelper.addMfeApps(data);
    return;
  } catch (error) {
    return error;
  }
}
```

註冊MFE詳細資訊

MFE已準備好進行測試。

測試自定義MFE

本地設定

1. 構建MFE應用程式。
2. 將角度應用程式分佈裝入為相應微服務服務的容器的資源資料夾中(例如，/root/MFE-app/dist/:/home/node/app/assets)。
3. 重新啟動microservice容器。
4. 刷新瀏覽器。

生成更改

1. 在映像建立過程中，更新Dockerfile以將Angular應用程式分發打包到容器內的assets資料夾中。側面導航顯示測試MFE選單。
2. 按一下Test MFE選單載入MFE(URL https://<host>/ex/test-mfe)。

附註：載入新MFE時，不會顯示側面導航，因為它未整合到新MFE中。

啟用側面導航

1. 在test-mfe/src/types/decl.d.ts路徑中建立新文件。
2. 新增以下宣告行：

```
declare module '@cisco-bpa-platform/utility-state';
```

3. 開啟"test-mfe/extra-webpack.config.js"並新增`config.externals = ['@cisco-bpa-platform/utility-state']`。

上述步驟定義@cisco-bpa-platform/utility-state並跳過匯入錯誤。

4. 驗證並安裝以下依賴項：
    - "@angular/cdk":"~14.2.7"
    - "@angular/材料":"~14.2.5"
    - "@angular/本地化":"^14.2.6"
    - "@cisco-bpa-platform/cxui-components":"4.0.1-1003 "
    - "@cisco-bpa-platform/data-access-services":"4.1.1-1003 "
    - "@cisco-bpa-platform/login":"4.0.3-1002 "
    - "@cisco-bpa-platform/shared-library":"4.0.1-1001 "
    - "@cisco-bpa-platform/spogui-components":"4.1.1-1005 "
    - "@cisco-bpa-platform/ui-shared-components":"4.1.1-1005 "
    - "@cisco-bpa-platform/util":"4.0.1-1002 "
    - "@ngrx/component-store":"~14.0.0"
    - "@ngrx/effects":"~14.0.0"
    - "@ngrx/實體":"~14.0.0"
    - "@ngrx/router-store":"~14.0.0"
    - "@ngrx/store":"~14.0.0"
    - "檔案保護程式":"^2.0.5"
    - 開發依賴項
    - "@types/file-saver":"^2.0.7"

附註：必須使用所有cisco-bpa-platform庫的正確版本。

---

5. 按如下所示更新以下根級別檔案：

- “app.component.html”

- "app.component.ts"

```
import { Component, HostListener } from @angular/core';
import { Router } from @angular/router';
import { Observable } from 'rxjs';
import { Loaderservice } from '@cisco-bpa-platform/data-access-services/auth';
@Component({
    selector: 'app-test-mfe-root',
    templateUrl : '. /app. component. html',
    styleUrls: ['./app.component.scss']
})
export class AppComponent {
    title = 'test-mfe';
    showLoader = false;
    loadingSpinner$: Observable
```

```
        = this.loaderService.spinnerObsv$; leftNavWidth = ['50px', '250px']; isExpanded = false; i
```

- "app.module.ts"

```
import { HTTP_INTERCEPTORS, HttpClientModule } from '@angular/common/http';
import { NgModule } from '@angular/core' ;
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { LoaderModule } from '@cisco-bpa-platform/cxui-components/loader';
import { AuthEffects, Authlnterceptor, authReducer } from '@cisco-bpa-platform/login/feature-logir
import { SideNavModule } from '@cisco-bpa-platform/spogui-components/side-nav';
import { EffectsModule } from '@ngrx/effects';
import { StoreModule } from '@ngrx/store';
import { environment } from '../environments/environment';
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';

@NgModule({
    declarations : [AppComponent],
    imports: [BrowserModule, AppRoutingModule, HttpClientModule,
        BrowserAnimationsModule,
        HttpClientModule,
        StoreModule. forRoot({ ['auth']: authReducer }, {
            runtimechecks: {
                strictStatelmmutability: false,
                strictActionlmmutability: false
            }
        }),
        EffectsModule.forRoot([AuthEffects]), LoaderModule, SideNavModule],
    providers: [
        { provide: ‘config’, useValue: { apiUrl: environment .apiUrl, mfa: environment .mfa } },
        {
            provide: HTTP_INTERCEPTORS,
```

```
            useClass: AuthInterceptor,
            multi: true,
        },
    ],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

- "environment.prod.ts"

```
const URL = `${i window.location.href.substring(0, window.location.href.lastIndexOf('ex'))}bpa` ;
export const environment = {
    production: true,
    appTitle: "BPA | ",
    appName: "Business Process Automation",
    apiurl: `${URL}`,
    assetsUrl: "./assets",
    packageUr1: "./assets/packages",
    mfa: {
        enabled: false,
        o t p_u r 1 : 'https://ssologon-prd.sm.bankofamerica.com/ssoconnect/sendOTP.htm1'
    },
    sync: {
        groups: false,
        users: false
    },
    autoRef reshInterval : 60000,
    showFormBuilderListItems: false,
    deploymentType: "not_on_prem"
};
```

- "環境.ts"

```
const HOSTNAME = window.location.hostname;
const URL = `http ://${HOSTNAME}: 8000`;
export const environment = {
    production: false,
    appTitle: "BPA | ",
    appName: "Business Process Automation",
    apiUrl: `${URL}`,
    packageUrl: `http://${HOSTNAME}:9090/assets/packages`,
    mfa : {
        enabled: false,
        otp_url: 'https://ssologon-prd.sm.bankofamerica.com/ssoconnect/sendOTP.html'
    },
    sync: {
        groups: false,
        users: false
    },
    autoRefreshinterval: 60000,
    showFormBuilderListItems: false,
    deploymentType: "not_on_prem"
};
```

側面導航已準備好進行測試。

在MFE中新增新路由

1. 生成comp-1和comp-2元件。
2. 在路由檔案中匯入這些元件。

```
import { CompOneComponent } from './comp-one/comp-one.component';
import { CompTwoComponent } from './comp-two/comp-two. component';
```

3. 定義字首路徑。

```
const includePath = window.location.pathname.substring(1, window.location.pathname.lastIndexOf('/ex'));
let prefixPath = 'ex';
if (includePath.length > 1) {
    prefixPath = `${includePath}/${prefixPath}`
}
```

4. 定義路由。將預設路由設定為最後路由。

```
const routes: Routes = [
    { path: `$iprefixPath}/test-mfe/compone`, component: CompOneComponent },
    { path: `${prefixPath}/test-mfe/comptwo`, component: CompTwoComponent },
    { path: '**', component: EmptyRouteComponent }
];
```

5. 檢視完整代碼。

```
import { APP_BASE_HREF } from '@angular/common';
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { EmptyRouteComponent } from './empty-route/empty-route.component';
import { CompOneComponent } from './comp-one/comp-one.component';
import { CompTwoComponent } from './comp-two/comp-two.component';

const includePath = window.location.pathname.substring(1, window.location.pathname.lastIndexOf('/ex'));
let prefixPath = 'ex';
if (includePath.length > 1) {
  prefixPath = `${includePath}/${prefixPath}`
}
const routes: Routes = [
  { path: `${prefixPath}/test-mfe/compone`, component: CompOneComponent },
  { path: `${prefixPath}/test-mfe/comptwo`, component: CompTwoComponent },
  { path: '**', component: EmptyRouteComponent }
];

You, 5 minutes ago | 1 author (You)
@NgModule({
  imports: [RouterModule.forRoot(routes, { enableTracing: true })],
  exports: [RouterModule],
  providers: [
    { provide: APP_BASE_HREF, useValue: '/' },
  ],
})
export class AppRoutingModule { }
```

審閱代碼

呈現自定義應用程式的巢狀選單

可以從自定義應用程式呈現巢狀選單。本節概述了準備MFE註冊文檔所需的步驟。為此引入了以下屬性：

- 級別：接受數字，從1開始；數量根據選單數量和級別增加而增加；較低級別應首先在BPA中註冊
- 父項：儲存應在其下顯示當前選單的父display_name屬性的值

為測試MFE自定義MFE應用程式建立的巢狀選單示例：

巢狀選單

要呈現巢狀選單：

1. 為級別、路徑和父屬性新增正確的值。

```
"layout_enabled": true,
```

```
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"display_name": "Test MFE",
"menu_path": "Application",
"custom": true,
"default": false,
"level": 1,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

如果選單包含子選單，則不應存在path屬性。級別1表示應首先註冊此選單。

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"display_name": "Test MFE",
"menu_path": "Application",
"custom": true,
"default": false,
"level": 1,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

在上圖中，父代值為測試MFE;因此，Test MFE Level One選單巢狀在Test MFE選單下。

第2級

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/compone",
"display_name": "Test MFE One",
"menu_path": "Application",
"parents": "Test MFE",
"custom": true,
"default": false,
"level": 2,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

第4級

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/comptwo",
"display_name": "Test MFE Two",
"menu_path": "Application",
```

```
"parents": "Test MFE",
"custom": true,
"default": false,
"level": 4,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

第5級

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/compone",
"display_name": "Test MFE First one",
"menu_path": "Application",
"parents": "Test MFE Level One",
"custom": true,
"default": false,
"level": 5,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

Test MFE First One選單巢狀在Test MFE Level One選單下。

第6級

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/comptwo",
"display_name": "Test MFE First Two",
"menu_path": "Application",
"parents": "Test MFE Level One",
"custom": true,
"default": false,
"level": 6,
"distribution_path" : "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

Test MFE First Two選單巢狀在Test MFE Level One選單下。

在步驟1中準備「mfe.json」檔案後，重新啟動microservice container以註冊條目。

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。