# React微前端与FDC3

2025 年 2 月 24 日•内容由 AI 生成，不能完全保障真实

写一个host app和一个client app，其中client app中有两个页面，一个search 一个detail，在search页面输入id ，在client页面显示其具体信息，client app作为微前端，注册到host app中，用到react技术和fdc3 ，还有nx federation

以下是一个简化的示例，展示如何使用 React、FDC3、NX Federation 创建一个包含 Host App 和 Client App 的微前端架构。假设你已经安装了 Node.js 和 npm，并且对 React 和 NX 有基本的了解。

### **1. 创建 NX 工作区**

首先，使用 NX CLI 创建一个新的工作区：

收起

bash

npx create-nx-workspace@latest my - micro - frontends --preset=reactcd my - micro - frontends

### **2. 创建 Host App**

在 NX 工作区中创建 Host App：

收起

bash

nx generate @nx/react:app host - app

### **3. 创建 Client App**

在 NX 工作区中创建 Client App：

收起

bash

nx generate @nx/react:app client - app

### **4. 配置 NX Federation**

在nx.json文件中配置 Module Federation。以下是一个简化的示例：

收起

json

{

"targets": {

"host - app": {

"architect": {

"webpack": {

"builder": "@nx/react:webpack",

"options": {

"outputPath": "./dist/apps/host - app",

"main": "./apps/host - app/src/main.tsx",

"tsConfig": "./apps/host - app/tsconfig.app.json",

"assets": [

"./apps/host - app/src/assets"

],

"styles": [

"./apps/host - app/src/styles.css"

],

"scripts": [],

"webpackConfig": {

"path": "./apps/host - app/webpack.extra.js"

}

},

"configurations": {

"production": {

"fileReplacements": [

{

"replace": "./apps/host - app/src/environments/environment.ts",

"with": "./apps/host - app/src/environments/environment.prod.ts"

}

],

"optimization": true,

"outputHashing": "all",

"sourceMap": false,

"extractCss": true,

"namedChunks": false,

"extractLicenses": true,

"vendorChunk": false,

"budgets": [

{

"type": "initial",

"maximumWarning": "2mb",

"maximumError": "5mb"

}

]

}

}

},

// 其他配置...

}

},

"client - app": {

"architect": {

"webpack": {

"builder": "@nx/react:webpack",

"options": {

"outputPath": "./dist/apps/client - app",

"main": "./apps/client - app/src/main.tsx",

"tsConfig": "./apps/client - app/tsconfig.app.json",

"assets": [

"./apps/client - app/src/assets"

],

"styles": [

"./apps/client - app/src/styles.css"

],

"scripts": [],

"webpackConfig": {

"path": "./apps/client - app/webpack.extra.js"

}

},

"configurations": {

"production": {

"fileReplacements": [

{

"replace": "./apps/client - app/src/environments/environment.ts",

"with": "./apps/client - app/src/environments/environment.prod.ts"

}

],

"optimization": true,

"outputHashing": "all",

"sourceMap": false,

"extractCss": true,

"namedChunks": false,

"extractLicenses": true,

"vendorChunk": false,

"budgets": [

{

"type": "initial",

"maximumWarning": "2mb",

"maximumError": "5mb"

}

]

}

}

},

// 其他配置...

}

}

}}

在apps/client - app/webpack.extra.js中配置 Client App 作为远程模块：

收起

javascript

const ModuleFederationPlugin = require('@module - federation/enhanced').ModuleFederationPlugin;

module.exports = {

plugins: [

new ModuleFederationPlugin({

name: 'client\_app',

filename: 'client - app.js',

exposes: {

'./Search': './apps/client - app/src/components/Search.tsx',

'./Detail': './apps/client - app/src/components/Detail.tsx'

},

shared: {

react: { singleton: true, eager: true },

'react - dom': { singleton: true, eager: true }

}

})

]};

在apps/host - app/webpack.extra.js中配置 Host App 消费 Client App：

收起

javascript

const ModuleFederationPlugin = require('@module - federation/enhanced').ModuleFederationPlugin;

module.exports = {

plugins: [

new ModuleFederationPlugin({

name: 'host\_app',

remotes: {

client\_app: 'client\_app@http://localhost:4201/client - app.js'

},

shared: {

react: { singleton: true, eager: true },

'react - dom': { singleton: true, eager: true }

}

})

]};

### **5. 实现 Client App 页面**

在apps/client - app/src/components/Search.tsx中：

收起

jsx

import React, { useState } from'react';import { useFDC3 } from '@finos/react - fdc3';

const Search = () => {

const [id, setId] = useState('');

const fdc3 = useFDC3();

const handleSearch = async () => {

try {

await fdc3.broadcast({ type: 'id - searched', id });

} catch (error) {

console.error('Error broadcasting id:', error);

}

};

return (

<div>

<input

type="text"

placeholder="Enter ID"

value={id}

onChange={(e) => setId(e.target.value)}

/>

<button onClick={handleSearch}>Search</button>

</div>

);};

export default Search;

在apps/client - app/src/components/Detail.tsx中：

收起

jsx

import React from'react';import { useFDC3 } from '@finos/react - fdc3';

const Detail = () => {

const fdc3 = useFDC3();

const [detail, setDetail] = useState(null);

React.useEffect(() => {

const subscription = fdc3.addContextListener('id - searched', (context) => {

// 模拟根据ID获取详细信息

const mockDetails = { id: context.id, name: `Name for ${context.id}` };

setDetail(mockDetails);

});

return () => {

subscription.unsubscribe();

};

}, [fdc3]);

return (

<div>

{detail && (

<div>

<p>ID: {detail.id}</p>

<p>Name: {detail.name}</p>

</div>

)}

</div>

);};

export default Detail;

### **6. 集成到 Host App**

在apps/host - app/src/App.tsx中：

收起

jsx

import React from'react';import type { ReactElement } from'react';import { lazy, Suspense } from'react';

const Search = lazy(() => import('client\_app/Search'));const Detail = lazy(() => import('client\_app/Detail'));

const App: React.FC = () => {

return (

<div>

<h1>Host App</h1>

<Suspense fallback={<div>Loading...</div>}>

<Search />

<Detail />

</Suspense>

</div>

);};

export default App;

### **7. 运行应用**

分别启动 Host App 和 Client App：

收起

bash

nx serve host - app

nx serve client - app

上述代码创建了一个简单的微前端架构，Host App 消费 Client App 暴露的组件，Client App 的 Search 页面通过 FDC3 广播 ID，Detail 页面接收 ID 并显示模拟的详细信息。实际应用中，获取详细信息的逻辑应替换为真实的 API 调用。