**Microfrontend-single Spa**

Overview of Steps

* Set up the root configuration (orchestrator).
* Create the Vue microfrontend applications.
* Integrate the microfrontends with the root configuration.
* Set up routing and inter-app communication.
* Run and deploy the microfrontend system.

**Step 1:** Create the Root Configuration

The root configuration (orchestrator) manages and loads all the microfrontends.

Steps:

Generate the Root Config: Run the Single-SPA CLI:

**Code:**

npx create-single-spa

if we want to skip git install

npx create-single-spa –skip-git

Project type: root-config.

Directory name: root-config.

Use the default options.

Navigate to the Project:

**cd root-config**

Install Dependencies: Install http-server to serve the root config:

npm install http-server

Start the Root Config: Serve the project:

bash

Copy code

npm start

This runs the root config at <http://localhost:9000>.

Step 2: Create Vue Microfrontend Applications

Each microfrontend is an independent Vue.js app.

Steps:

Generate a Microfrontend: Use the Single-SPA CLI to create a Vue microfrontend app:

  npm install -g @vue/cli

bash

Copy code

Project type: application / parcel.

Framework: vue.

Directory name: e.g., app-vue.

Accept the defaults for other options.

Repeat this process for additional microfrontends (e.g., app-vue-2).

Navigate to the Microfrontend Folder:

bash

Copy code

cd app-vue

Install Single-SPA Vue Adapter: Install single-spa-vue to make the Vue app compatible with Single-SPA:

bash

Copy code

npm install single-spa-vue

Modify main.js: Update the file to register the app with Single-SPA:

javascript

Copy code

import { h, createApp } from 'vue';

import singleSpaVue from 'single-spa-vue';

import App from './App.vue';

import router from './router';

const appOptions = {

  render: () => h(App),

  router,

};

const vueLifecycles = singleSpaVue({

  createApp,

  appOptions,

});

export const bootstrap = vueLifecycles.bootstrap;

export const mount = vueLifecycles.mount;

export const unmount = vueLifecycles.unmount;

Run the Microfrontend: Start the Vue app:

bash

Copy code

npm run serve

It will be served at <http://localhost:8080>.

Repeat these steps for additional microfrontends (e.g., app-vue-2).

Step 3: Connect Microfrontends to the Root Config

To load the microfrontends, you need to register them in the root configuration.

Steps:

Edit root-config/src/root-config.js: Register your microfrontends using Single-SPA’s registerApplication method:

javascript

Copy code

import { registerApplication, start } from "single-spa";

registerApplication({

  name: "@org-name/app-vue", // First microfrontend

  app: () => System.import("@org-name/app-vue"),

  activeWhen: ["/app-vue"], // Route to activate this app

});

registerApplication({

  name: "@org-name/app-vue-2", // Second microfrontend

  app: () => System.import("@org-name/app-vue-2"),

  activeWhen: ["/app-vue-2"], // Route to activate this app

});

start();

Update the Import Map: Modify the import-map.json file in the root-config’s public directory to include your microfrontends:

json

Copy code

{

  "imports": {

"@org-name/app-vue": "<http://localhost:8080/app.js>",

"@org-name/app-vue-2": "<http://localhost:8081/app.js>"

  }

}

Restart the Root Config: Stop and restart the root config server to load the updated import map:

bash

Copy code

npm start

Step 4: Add Routing in Microfrontends

If microfrontends need their own routing, set up vue-router in each Vue app.

Steps:

Install Vue Router:

bash

Copy code

npm install vue-router

Create a Router File (src/router/index.js):

javascript

Copy code

import { createRouter, createWebHistory } from 'vue-router';

import Home from '../views/Home.vue';

import About from '../views/About.vue';

const routes = [

  { path: '/', name: 'Home', component: Home },

  { path: '/about', name: 'About', component: About },

];

const router = createRouter({

  history: createWebHistory(), // Use web history for Single-SPA compatibility

  routes,

});

export default router;

Import the Router in main.js:

javascript

Copy code

import router from './router';

const appOptions = {

  render: () => h(App),

  router,

};

Step 5: Share Common Libraries

To optimize your system, share common dependencies like Vue between microfrontends.

Steps:

Update Import Map: Add Vue to the import-map.json in the root config:

json

Copy code

{

  "imports": {

"vue": "<https://unpkg.com/vue@3.2.0/dist/vue.esm-browser.js>"

  }

}

Mark Vue as a Peer Dependency: Update package.json in each microfrontend to mark Vue as a peer dependency:

json

Copy code

{

  "peerDependencies": {

    "vue": "^3.2.0"

  }

}

Step 6: Run the System

Start the root config:

bash

Copy code

cd root-config

npm start

Start each microfrontend:

bash

Copy code

cd app-vue

npm run serve

bash

Copy code

cd app-vue-2

npm run serve

Navigate to <http://localhost:9000> to see the root config. Use /app-vue and /app-vue-2 to load the microfrontends.

Step 7: Deploy the System

Deploy Each Microfrontend: Host each microfrontend separately using a CDN or hosting service (e.g., AWS S3, Netlify, or Vercel).

Update Import Map in Production: Modify the import-map.json in the root config to point to the deployed URLs of the microfrontends:

json

Copy code

{

  "imports": {

"@org-name/app-vue": "<https://cdn.example.com/app-vue/app.js>",

"@org-name/app-vue-2": "<https://cdn.example.com/app-vue-2/app.js>"

  }

}

Deploy the Root Config: Deploy the root config to a CDN or hosting service.

Optional Step: Enable Communication Between Microfrontends

If microfrontends need to share data or communicate:

Use an event bus (e.g., mitt):

javascript

Copy code

import mitt from 'mitt';

export const EventBus = mitt();

Use a shared state management library like Pinia or Vuex for global state.

By following these steps, you’ll have a fully functioning microfrontend system with Vue.js using Single-SPA. Let me know if you need help with a specific part!

has context menu