

微服务Portal界面集成方案（前端部分）

微服务Portal界面集成方案（前端部分）

- 关于本文档
- 名词约定
- 界面集成原理
- 界面集成流程
- 界面集成示例
- 其他实现
- 其他规范
 - 防JS冲突规范
 - 防本地存储溢出规范
 - 防本地存储冲突规范
 - 跨子应用数据交互规范
 - 链接跳转规范
 - 全局样式规范
 - SSO权限接口
- 示例工程界面截图
- Portal集成界面效果图

关于本文档

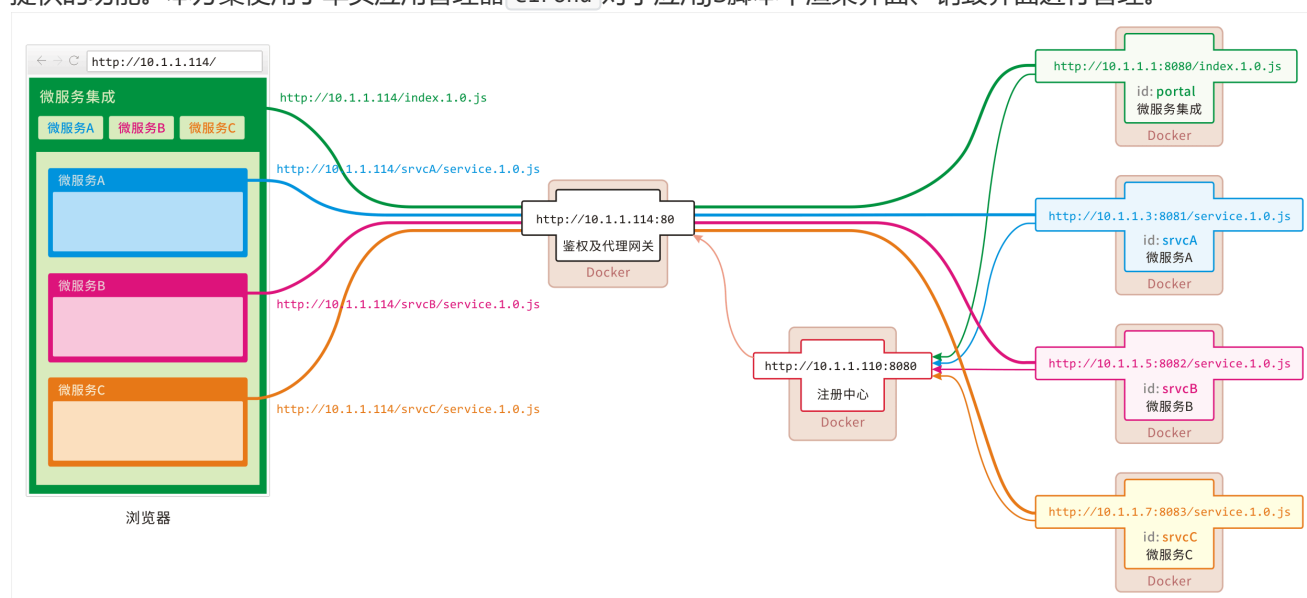
- 本文档提供一种以前端JS开发技术为主的微服务界面集成方案。
- 本文档面向有单页WebApp开发经验的前端开发人员，要求有 `Vue.js` 或 `React` 或 `Angular` 开发经验，有 `Webpack` 编译/打包JS经验。
- 对于不同JS框架编写的单页应用在同一页面上共存，本方案使用到单页应用管理器 `elrond` 来管理，它管理的子应用应该实现 `bootstrap`、`mount`、`umount` 三个生命周期。参见示例工程中的代码。

名词约定

子界面：微服务提供的JS可以响应特定的事件后渲染子应用功能所需界面。这些JS所渲染的界面，下面称之为 **子界面**。**子应用**：微服务提供的JS可以完成界面渲染，并响应界面上的交互，完成该微服务所要提供的功能。这些JS所提供的功能，下面称之为 **子应用**。

界面集成原理

如下图所示，在浏览器中访问入口页面时，该页面载入由不同的微服务提供的JS文件，这些JS脚本执行后，监听地址栏hash的变化，在监听到特定的hash值时，渲染微服务的前端界面，并响应界面上的交互，完成该微服务所要提供的功能。本方案使用了单页应用管理器 `elrond` 对子应用JS脚本中渲染界面、销毁界面进行管理。



前端技术兼容性：本方案支持各个子应用选择不同MVVM框架。支持React、Angular、Vue.js、Svelte、Preact等前端框架，也支持原生JS。通过这些MVVM框架的和JS/CSS模块化机制开发各个子应用，并基于Webpack打包成单独的JS文件。入口页面的子应用管理脚本通过监听路由变化，执行子应用生命周期函数，将不同子界面的视图绑定到不同的DOM节点，从而实现在同一个界面中不同前端框架编写的子应用共存。

界面集成流程

1. 准备子应用JS

- 子应用所对应的JS对象，对外只提供三个方法：初始化方法（`bootstrap`）、界面渲染方法（`mount`）、界面销毁方法（`umount`），由 `elrond` 管理子应用的初始化、渲染、销毁。
- 使用 `Vue.js`、`React`、`Angular` 或原生JS，编写单页应用
- 子应用对应的JS对象在调用 `mount()` 时，对前一步编写的单页应用进行初始化，根据这个单页应用使用的JS框架不同，处理的方式也有区别。
 - 如果单页应用是基于 `Vue` 的，会实例化一个指定了 `el` 属性的 `Vue` 对象
 - 如果单页应用是基于 `React` 的，会创建一个 `React` 组件并调用 `ReactDOM.render()` 渲染这个组件到页面上
 - 如果单页应用是基于 `Angular` 的，会将 `template` 对应的HTML写入到页面上，然后执行 `bootstrapModule()`
- 子应用对应的JS对象在调用 `umount()` 时，对前一步编写的单页应用进行销毁，根据这个单页应用使用的JS框架不同，处理的方式也有区别。
- 如果单页应用是基于 `Vue` 的，会调用上一步实例化的 `Vue` 对象的 `$destroy()` 方法
- 如果单页应用是基于 `React` 的，会调用 `ReactDOM.unmountComponentAtNode()` 方法销毁应用相关DOM元素
- 如果单页应用是基于 `Angular` 的，会调用上一步实例化的 `Angular` 组件的 `destroy()` 方法
- 配合使用 `Webpack` 将界面呈现所涉及的HTML、CSS、图标等资源都打到单页应用JS里

2. 准备子应用相关微服务

- 使用 `Java`、`Nodejs` 或其他后端技术，实现RESTful风格的后端接口，上一步编写的子应用可调用这些接口与后端进行交互

- 使用 `Java`、`Nodejs` 或其他后端技术，实现微服务注册功能——向微服务注册中心提供以下配置
 - 子应用的id
 - 子应用的JS路径
 - 子应用要注册的菜单名
 - 子应用要集成到Portal界面的widget对应的JS路径

3. 运行注册中心服务

- 使用 `Java`、`Nodejs` 或其他技术，实现微服务注册接口、微服务发现接口。接受其他微服务的注册，接受集成应用对其他子应用的配置的查询
- 使用 `Java`、`Nodejs` 或其他技术，启动Web服务器，供其他微服务访问
- 用 `docker` 打包微服务的运行环境为镜像，并在服务器上运行

4. 运行鉴权及代理网关服务

- 为了统一鉴权、统一负载均衡、解决跨域限制等等。有必要实现鉴权服务、代理网关服务，使用 `Java`、`Nodejs` 或其他技术，实现鉴权服务、代理网关服务。对所有微服务的访问，都由代理网关转发。
- 使用 `Java`、`Nodejs` 或其他技术，启动Web服务器，供用户的浏览器访问前端JS及后端接口
- 用 `docker` 打包微服务的运行环境为镜像，并在服务器上运行

5. 运行各子应用相关微服务

- 使用 `Java`、`Nodejs` 或其他技术，启动Web服务器，供用户的浏览器访问前端JS及后端接口
- 用 `docker` 打包微服务的运行环境为镜像，并在服务器上运行

界面集成示例

本示例前端以Vue框架为例示例子应用开发的要点。

1. 约定子应用所属的微服务的id

这个id用来识别微服务，在微服务的前后端编程过程中经常要用到，非常重要。id必须由字母和数字组成，不能使用特殊符号，不要和其他已有微服务重名，一定要询问Portal平台管理员id是否可用。

2. 创建子应用Vue工程

本示例中Vue工程目录结构如下

```

./
├── package.json           前端工程配置文件
├── server.js              启动http服务，使用js可被访问，向注册中心注册子应用id、js路径等信息
├── webpack.config.js      webpack配置文件
├── dist                  js发布目录
│   ├── service.d97144b0.js  webpack编译得到的子应用的JS
│   └── widget.42c41e21.js   webpack编译得到的Portal小部件的JS
├── assets                资源目录，存放字体、图片等比较没有打包到js中的资源
└── src                  js源码目录
    ├── config.js          微服务id存放在这个文件中，webpack、server等js都有读取这个id配置
    ├── routes.js          路由配置
    ├── service.js         子应用
    ├── service.register.js 注册子应用到elrond
    ├── subroute1.component.vue 响应路由切换的vue组件
    ├── subroute2.component.vue 响应路由切换的vue组件
    └── subroute3.component.vue 响应路由切换的vue组件
  
```

widget.js

Portal部件的JS

widget.register.js

注册子应用的Portal小部件到elrond

3. 引入前端工程依赖的js包

本子应用依赖 webpack, babel-core, babel-loader, vue-loader, vue-template-compiler, vue, vue-router, elrond-spa-vue等包, 请参考示例工程, 使用 yarn 或 npm 安装依赖。

4. 配置id

本子应用id命名为 form, config.js 内容如下

```
module.exports = {  
  // 配置微服务应用的id, 只允许使用字母和数字, 注意要不要和其他微服务id重复  
  SERVICEID : 'form'  
}
```

微服务id存放在这个文件中, webpack.config.js、server.js 及大部分涉及路由及资源的js都要读取 config.js 中这个id配置

5. 编写子应用js

子应用js文件命名为 service.js, js内容十分简单:

- 配置了一个Vue实例, 这个Vue实例主要是渲染一个路由视图
- 使用 elrond-spa-vue 包裹Vue配置, 以便 elrond 可以管理这个Vue实例
- 返回有三个方法 (bootstrap、mount、unmount) 的对象, 以便在 service.register.js 里注册Vue实例为子应用

```
import {SERVICEID} from './config.js'  
import Vue from 'vue/dist/vue.min.js'  
import VueRouter from 'vue-router'  
import elrondSpaVue from './elrond-spa-vue'  
import routes from './routes'  
import axios from 'axios'  
  
axios.defaults.baseURL = `/${SERVICEID}`  
Vue.use(VueRouter)  
const router = new VueRouter({  
  routes  
})  
  
const vueLifecycles = elrondSpaVue({  
  Vue,  
  appOptions: {  
    router,  
    el: `#${SERVICEID}`,  
    template: `      <router-view :key="$route.fullPath"></router-view>  
    </div>`  
  }  
})  
  
export function bootstrap (props) {  
  return vueLifecycles.bootstrap(props)
```

```

}

export function mount (props) {
  return vueLifecycles.mount(props)
}

export function unmount (props) {
  return vueLifecycles.unmount(props)
}

```

注意，

6. 编写子应用入口js

前一步编写的子应用，还需要在 `elrond` 正式注册，`service.register.js` 内容如下

```

import {SERVICEID} from './config.js'
import * as service from './service.js'
window.elrondSpa.registerApplication(`service-${SERVICEID}`, service, _ => {
  return window.location.hash.startsWith(`#${SERVICEID}`)
})

```

7. 配置路由及编写路由对应的子组件

本示例中 `routes.js` 内容如下

```

import {SERVICEID} from './config.js'

export default [
  { path: `/${SERVICEID}/edit`, component: _ => import('./subroute1.component.vue') },
  { path: `/${SERVICEID}/search`, component: _ => import('./subroute2.component.vue') },
  { path: `/${SERVICEID}/stat`, component: _ => import('./subroute3.component.vue') },
]

```

注意：上面的写法里子应用的vue子组件使用异步载入，以减少子应用生成的 `service.js` 的体积，如果你的子应用打包后 `service.js` js 体积很小，和Portal平台管理员沟通，获得许可后，可以不使用js异步加载

注意：经测试页面引入 `zone.js` 后，`zone.js` 改写了Promise，会引起组件异步载入失败。即使用 `zone.js` 后不能再使用异步载入js功能。

```

import {SERVICEID} from './config.js'
import Subroute1 from './subroute1.component.vue'
import Subroute2 from './subroute2.component.vue'
import Subroute3 from './subroute3.component.vue'

export default [
  { path: `/${SERVICEID}/edit`, component: Subroute1 },
  { path: `/${SERVICEID}/search`, component: Subroute2 },
  { path: `/${SERVICEID}/stat`, component: Subroute3 },
]

```

本示例中 `subroute1.component.vue` 内容如下

```

<template>
  <div>
    响应路由变化，渲染了Vue子组件 1 ！
  </div>
</template>

<script>
import {SERVICEID} from './config.js'
export default {
  data(){
    return {
      SERVICEID
    }
  }
}
</script>

```

8. 编写向注册中心注册子应用的功能

前提条件

- 从前面提到的 `config.js` 中获取子应用id。本示例中子应用id为 `form`
- 使用技术手段获取注册中心的子应用注册接口的url。本示例中注册接口url为 `http://10.1.1.110:8080/service`
- 使用技术手段获取编译后子应用JS的文件名，获取JS在微服务中对应的url。本示例中JS的url为 `http://10.1.2.3:8001/service.jiag2c00.js`
- 使用后端技术，向注册接口，提交子应用的配置，本示例子应用配置JSON数据如下

```

{
  "id": "form",
  "name": "报关单",
  "version": "0.1.0",
  "framework": "Vue",
  "url": "http://10.1.2.3:8001",
  "serviceJS": "http://10.1.2.3:8001/service.jiag2c00.js",
  "menus": [{
    "id": "form",
    "name": "报关单",
    "children": [{
      "id": "form-edit",
      "name": "报关单录入",
      "link": "/form/edit"
    }, {
      "id": "form-search",
      "name": "报关单检索",
      "link": "/form/search"
    }, {
      "id": "form-stat",
      "name": "报关单统计",
      "link": "/form/stat"
    }
  ]
}

```

```

    }
  ],
  "widgets": [{
    "id": "widget-form-todos",
    "name": "待处理报关单",
    "widgetJS": "http://10.1.2.3:8001/widget.jiag2c00.js",
    "colSpan": 1,
    "maxHeight": 200,
  }
],
"healthCheck": {
  "path": "http://10.1.2.3:8001/healthCheck"
}
}

```

注意：

- 子应用JS、Portal部件（widget）JS需要给出全路径
- 每个子应用可以注册多个菜单，可以注册一级菜单，也可以注册二级菜单。相同id的一级菜单将会合并为一个
- Portal界面为3列
- 由于Portal界面的位置有限，每个子应用暂时只允许向Portal界面添加一个widget
- 为了一屏至少可以显示9个widget，要求占用两列或三列宽的widget需要管理员审核才能生效
- 为了一屏至少可以显示9个widget，widget的高度超过240部分将被隐藏

9. 实现子应用界面集成微服务

• 前端部分

子应用界面集成微服务前端部分文件如下

bootstrap.css	本案例要求各子应用统一使用Bootstrap内置字体图标
index.css	集成页面CSS。主要定义一些标准样式供子应用使用以保持统一风格。
index.html	集成页面
index.js	集成页面JS。主要功能获取子应用js配置，根据配置渲染导航菜单
elrond-spa.js	单页应用管理器elrond
zone.js	为防止zone.js重复引入报错，要求Angular编写的子应用不再引入zone.js

index.html 内容如下，注意接口 `/declareChildApplication` 以jsonp的格式返回所有子应用的配置。

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>示例微服务的界面集成</title>
  <link rel="stylesheet" href="bootstrap.css">
  <link rel="stylesheet" href="index.css">
</head>
<body>
  <div class="sidebar">
    <h4 class="tac">微服务界面集成演示</h4>

```

```

    <div class="menus-container" id="menus-container">
    </div>
</div>
<div id="widgets-container" class="widgets-container"></div>
<div id="services-container" class="services-container"></div>
<script src="elrond-spa.js"></script>
<script src="index.js"></script>
<script src="/declareChildApplication?callback=onServicesConfigLoaded"></script>
</body>
</html>

```

index.js 关键内容如下

```

(function () {
  function loadJS(url) {
    return new Promise(function (resolve, reject) {
      var s = document.createElement('script')
      s.setAttribute('src', url)
      document.head.appendChild(s)
      s.onload = resolve
      s.onerror = reject
    })
  }

  window.onServicesConfigLoaded = function (servicesConfig) {
    let servicesJS = []
    let widgetsHtml = []
    let servicesHtml = []
    let navHtml = [`<ul class="menu level1">
    <li class="menu-item level1">
    <a href="#" data-route-path="/">首页</a>
    </li>
    </ul>`]

    for (let i = 0; i < servicesConfig.length; i++) {
      let widgets = servicesConfig[i].widgets
      if (widgets && widgets.length) {
        for (let j = 0; j < widgets.length; j++) {
          widgetsHtml.push('<div id="' + widgets[j].id + '" class="' + widgets[j].id + '">
</div>')
          servicesJS.push(widgets[j].widgetJS)
        }
      }
    }

    for (let i = 0; i < servicesConfig.length; i++) {
      if (servicesConfig[i].serviceJS) {
        servicesHtml.push('<div id="' + servicesConfig[i].id + '" class="' +
servicesConfig[i].id + '"></div>')
        servicesJS.push(servicesConfig[i].serviceJS)
      }
      let menus = servicesConfig[i].menus
      if (menus && menus.length) {

```



```

navHtml.push('<ul class="menu level1">')
for (let j = 0; j < menus.length; j++) {
  let submenus = menus[j].children
  if (!submenus || !submenus.length) {
    navHtml.push('<li class="menu-item level1">')
    navHtml.push('<a>${menus[j].name}</a>')
    navHtml.push('</li>')
  } else {
    navHtml.push('<li class="menu-item level1 haschilds">')
    navHtml.push('<a class="haschilds">${menus[j].name}</a>')
    navHtml.push('<ul class="menu level2">')
    for (let k = 0; k < submenus.length; k++) {
      navHtml.push('<li class="menu-item level2">')
      navHtml.push('<a href="#${submenus[k].link}" data-route-')
      navHtml.push('path="${submenus[k].link}">${submenus[k].name}</a>')
      navHtml.push('</li>')
    }
    navHtml.push('</ul>')
    navHtml.push('</li>')
  }
}
navHtml.push('</ul>')
}

navHtml = navHtml.join('').replace(/\n/g, '')
widgetsHtml = widgetsHtml.join('').replace(/\n/g, '')
servicesHtml = servicesHtml.join('').replace(/\n/g, '')
document.getElementById('menus-container').innerHTML = navHtml
document.getElementById('widgets-container').innerHTML = widgetsHtml
document.getElementById('services-container').innerHTML = servicesHtml

var loadJSPromise = []
for (let i = 0; i < servicesJS.length; i++) {
  loadJSPromise.push(loadJS(servicesJS[i]))
}
Promise.all(loadJSPromise).then(window.elrondSpa.start)
}
})();

```

- 后端部分

实现接口 `/declareChildApplication` 以jsonp的格式返回所有子应用的配置。
具体实现不在《微服务Portal界面集成方案前端部分》说明范围内。

10. 实现注册中心

注册中心的实现不在《微服务Portal界面集成方案前端部分》说明范围内。

11. 实现鉴权网关

鉴权网关的实现不在《微服务Portal界面集成方案前端部分》说明范围内。

12. 实现代理网关

代理网关的实现不在《微服务Portal界面集成方案前端部分》说明范围内。

其他实现

- Vue实现子应用的前后端代码详细实现，代码示例工程源码
- React、Angular、及原生JS分别实现子应用，代码示例工程源码
- Portal界面集成页面，代码示例工程源码
- 注册中心、代理网关的后端实现，代码示例工程源码

其他规范

防JS冲突规范

1. 为了防止子应用冲突，所有子应用不要直接创建全局方式，不要创建全局变量
2. 子应用创建的方法或变量，都要放到子应用ID的命名空间下，例如

```
import {SERVICEID} from './config.js'
let ns = window[SERVICEID] = window[SERVICEID] || {}
ns.foo = { bar: 'val1', baz: 'val2' }
ns.qux = (a, b) => {}
```

防本地存储溢出规范

1. Chrome下每个域名只能存50个 cookie，最多4K数据。
为了防止cookie存储溢出，普通子应用不要用 cookie 存储数据（系统后台会清掉各子应用写的 cookie 来强制禁止使用cookie），
特殊情况经与平台管理员联系后可开放有限的 cookie 写权限。
2. Chrome下 sessionStorage localStorage 的容量为5M。超过5M后将无法写入。
3. 各子应用尽量用 sessionStorage 代替 localStorage，防止 localStorage 膨胀
4. 各子应用不要向 localStorage 存储大量数据（超过100K）

防本地存储冲突规范

1. 为了防止子应用冲突，所有子应用向 localStorage 里读写数据时，必须带上子应用ID为前缀例如

```
import {SERVICEID} from './config.js'
localStorage.setItem(`${SERVICEID}.foo`, 'value')
localStorage.setItem(`${SERVICEID}.foo.bar`, 'value')
```

跨子应用数据交互规范

1. 在 index.js 里我们定义了 portal.global 为一个 Observable 实例，Observable 是 订阅/发布模式 的实现，所以 portal.global 支持

```
get
set
subscribe
unsubscribe
```

等方法

```
function callback(value, path) {
  console.log(value, path);
}

portal.global.set('foo', { bar: 'value' }) // 改变属性foo的值
portal.global.set('foo.bar', 'newValue') // 改变属性 foo.bar的值
portal.global.get('foo') // 获取属性foo 的值
portal.global.get('foo.bar') // 获取属性bar 的值
portal.global.subscribe('foo', callback) // 监听foo属性的改变
portal.global.subscribe('foo.bar', callback) // 监听bar属性的改变
portal.global.unsubscribe('foo', callback) // 停止监听 foo 属性的改变
portal.global.unsubscribe('foo.bar', callback) // 停止监听 bar 属性的改变
```

2. 子应用可以在 `bootstrap` 生命周期函数中声明自己的跨服务数据项。声明方式类似于：

```
portal.global.set("form.count",1)
```

3. 其他子界面可以在自己的bootstrap生命周期函数中声明要监听跨服务业数据项。声明方式类似于：

```
portal.global.subscribe("form.count",function(value, path){
  //获取value做相应动作
});
```

链接跳转规范

1. 子应用里的链接要设置属性 `target="_blank"`，使页面在新窗口打开。
2. 集成页面的js会遍历没有设置属性为 `target="_blank"` 的链接，并禁止这些链接在当前窗口打开。

全局样式规范

1. 在集成页面已经引入 `bootstrap.css`，并且会对 `Bootstrap` 样式作统一调整，因此所有子应该尽量使用 `Bootstrap` 样式，并且不要自己再引入 `bootstrap.css`。使界面风格保持一致。
2. `h1/h2/h3/h4/h5/h6` 样式表示1至6号标题。参见 `Bootstrap` 文档 <http://v3.bootcss.com/css/#type-heading>
3. `bg-primary/bg-info/bg-success/bg-danger/bg-warning` 样式表示不同情境的背景色。参见 `Bootstrap` 文档 <http://v3.bootcss.com/css/#helper-classes-backgrounds>
4. `text-muted/text-primary/text-info/text-success/text-danger/text-warning` 样式表示不同情境的文本颜色。参见 `Bootstrap` 文档 <http://v3.bootcss.com/css/#helper-classes-colors>
5. `btn-default/btn-primary/btn-info/btn-success/btn-danger/btn-warning` 样式表示不同情境的按钮颜色。参见 `Bootstrap` 文档 <http://v3.bootcss.com/css/#buttons-options>
6. `btn-xs/btn-sm/btn-md/btn-lg` 样式表示不同规格的按钮大小。参见 `Bootstrap` 文档 <http://v3.bootcss.com/css/#buttons-sizes>
7. `mg-xs/mg-sm/mg-md/mg-lg` 样式表示不同尺寸的空白（margin），`mg-l/mg-r/mg-t/mg-b` 表示左/右/上/下的空白。这不是来自Bootstrap，是本项目自定义的样式。
8. `pd-xs/pd-sm/pd-md/pd-lg` 样式表示不同尺寸的内补（padding），`pd-l/pd-r/pd-t/pd-b` 表示左/右/上/下的内补。这不是来自Bootstrap，是本项目自定义的样式。
9. `sz1/sz2/sz3/sz4/sz5/sz6` 样式从小到表示预定义的几种字号。这不是来自Bootstrap，是本项目自定义的样式。

SSO权限接口

要求SSO系统通过某个URL返回如下JSON数据：

```
{
  "username": "admin",
  "menus": [
    "form",
    "form-edit",
    "form-search",
    "form-stat"
  ],
  "widgets": [
    "form-stat-date",
    "form-stat-type"
  ]
}
```

- 其中menus表示当前用户拥有哪些菜单项权限；widgets表示当前用户拥有Portal页面哪些部件权限。
- 如果当前用户未登录，则此URL需返回401状态码。
- 如果此URL返回401，则Portal将重定向到SSO的登录页面。

示例工程界面截图



微服务界面集成演示

◦ 首页

报关单

◦ 报关单录入

◦ 报关单检索

◦ 报关单统计

税收管理

贸易管理

系统配置

这儿是一个子应用，使用Vue编写。

列一	列二	列三	列四
view dnom	wbrmw pdcqnnhwgpd	s jxq c coapmscxa h	查看详情
wgeuoqvih	jc yh u ximwytofcvgo	ejxn rng hkflap j	查看详情
yeovj glwu	mhmsvb xvit v wr ho	rgb uxz xm uihdjqfi	查看详情
wrfibrvv	cu slbodf mwsknbn q	whv kkefkz zr sof	查看详情
zmvzqxt cu	mfvmfsxtcpjnezc	to b ke eo bw lauxlg k	查看详情

Portal集成界面效果图

微服务界面整合门户

◦ 首页

报关单

◦ 报关单录入

◦ 报关单检索

◦ 报关单统计

税收管理

◦ 税率管理

◦ 减免税管理

贸易管理

系统配置

按季报关单位数

9,280

按季已处理报关单

81,212

按季总涉税额

102,400

按季处理舱单

13,600

按工作类型统计

设置

Sales

Marketing

Development

Technology

Administration

Allocated Budget

Expected Spending

Actual Spending

按商品类型统计

设置

Industries

Technology

Forex

Gold

Forecasts

按时间统计

设置

Mon

Tue

Wed

Thu

Fri

Sat

Sun

报关单

设置

海关编号	海关口岸	申报单位	申报日期
2011180354730	外高桥关	上海华生国际货物运输代理有...	2017-10-18
2006051623674	浦东港办	广州市运达报关行黄埔分部	2017-10-18
2006051623675	浦东港办	广州市运达报关行黄埔分部	2017-10-18
2006051623676	浦东港办	广州市运达报关行黄埔分部	2017-10-18

报关单类型

请选择

预录入编号

海关编号

出口口岸

备案号

出口日期

选择出口日期

申报日期

选择申报日期

搜索

商品管理

设置

名称	地址	电话	添加时间
英尚外贸有限公司	湖南省长沙市岳麓区...	18887988987	2018-1-14
泽元外贸有限公司	河南省长沙市岳麓区...	13265748795	2018-1-14
纤云电子商务有限公司	海南省长沙市岳麓区...	18779543209	2018-1-14
纤云电子商务有限公司	海南省长沙市岳麓区...	18779543209	2018-1-14

减免税

设置

货号	货品名称	减免税率	政策文件
0101081001	机械探伤仪	15%	海关总署179号令《中华人民共和国海关进...
0101081001	机械探伤仪	15%	国内投资项目不予免税的进口商品目录（20...
0101081001	机械探伤仪	15%	《关于“十二五”期间在我国陆上特定地区开...
0101081001	机械探伤仪	15%	《关于调整重大技术装备进口税收政策的通...

报关单审核

设置

海关编号	申报单位	申报日期	审核状态
2011180354730	上海华生国际货物运输代理有限公司	2017-10-18	未审核
2006051623674	广州市运达报关行黄埔分部	2017-10-18	通过
2011180354730	上海华生国际货物运输代理有限公司	2017-10-18	未审核
2006051623674	广州市运达报关行黄埔分部	2017-10-18	通过

舱单

设置

舱单编号	收件公司	商品	发货地	收货人	电话
QWERT1	优衣库			张一山	010-34235345
TREWQ2	金利来			张一山	010-888888
ASDFG3	三星智能手机			李斯其	010-666666
KJHFGFG4	H&M			王文武	010-09824545

