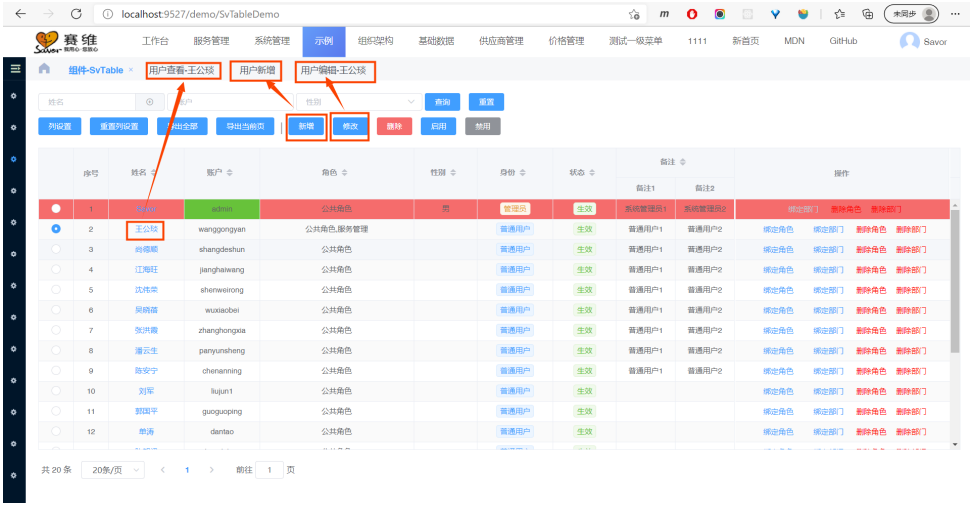
页签新增和关闭说明

实例展示

如图1所示，点击页面上的用户新增、查看、修改功能会分别打开新的系统页签，每个页签对应的对不同用户的不同操作

图1

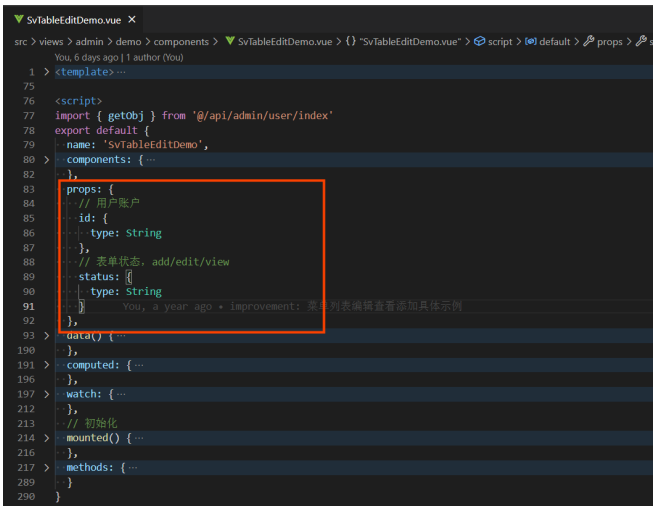


注意事项

图1中新增、查看、修改功能打开的是同一组件SvEditDemo.vue，只是在url中传入了不同的状态status（查看view，新增create，修改edit），在组件SvEditDemo.vue获取当前的status，通过判断status来控制组件中的内容展示

开发步骤

1. 创建SvEditDemo.vue组件，在组件的props中接收url中传递的参数，例如当前状态status，用户id等



2. 在SvEditDemo中通过props中的status来展示不同用户的不同状态对应的内容

3. 在路由router/index.js中配置SvEditDemo.vue组件的路由，需要配置以下内容：

3.1 **path**：路径SvTableEditDemo和参数

status：当前状态必传

pageId：当前页面编码pageId在调用$openTag方法的时候系统自动生成

id：用户id根据组件中是否需要决定是否传递，?代表id可传可不传。如果父组件有需要传递的其他参数例如name，也可以通过这种方式拼接到path中

title：页签名称必传

3.2 **prop**：**true**，path中的参数会作为props传入到组件SvTableEditDemo中

3.3 **meta**：路由缓存

noche：**true**，是否缓存页面，即切换页签的时候之前页面时之前的内容是否被清除，noche为true的时候代表页面会进行缓存，不清除

currentTab：对应的导航栏页签的code，即菜单配置中一级菜单的编码，例如导航栏中的示例页签为一级菜单示例，对应的code为demo，所以当前配置中currentTab为demo

layout：页面布局，通常来说都是用layout，带菜单和带标签栏的页面布局样式（其他的两种布局样式分别是，dashboard：布局中无菜单和页签，参考工作台页面；layoutWithoutSidebar：布局中无菜单的样式，参考测试一级菜单）

```
// SvTable功能，新增，修改，查看功能
{
  path: 'SvTableEditDemo/:status/:pageId/:id?/:title',
  component: () => import('@/views/admin/demo/components/SvTableEditDemo'),
  name: 'SvTableEditDemo',
  props: true, // 路由中的params会当作props传入组件中
  meta: {
    noCache: true, // 是否缓存，切换页签的时候不重新查找
    currentTab: 'demo', // 对应的导航栏页签的code(即一级菜单的编码)，本功能在示例下边，所以为demo
    layout: 'layout' // 布局样式选择带菜单和标签栏的layout类型
  }
}
]
}
```

4. 在父组件SvTableDemo中调用this.$openTag方法打开新的页签

4.1 新增，传入this和参数，name：路由名称，title：标签页名称，status：当前状态create（分别和router/index.js中path中配置的路由参数对应，同时也和子组件SvEditDemo中props接收的参数对应）

```
// 新增
handleAdd() {
  // 框架通用方法，打开新页签
  // name：router/index.js中配置的路由的name
  // title：标签名称
  // status：页签状态，自定义(当前文件中每个功能的status状态应该唯一)
  this.$openTag(this, {
    name: 'SvTableEditDemo',
    title: '用户新增',
    status: 'create'
  })
},
```

4.2 查看，传入this和参数，name：路由名称，title：标签页名称，status：当前状态view，id：当前用户id（分别和router/index.js中path中配置的路由参数对应，同时也和子组件SvEditDemo中props接收的参数对应）

```
// 查看
handleView(row) {
  // 是否选中一条数据
  if (row) this.currentRow = row
  if (!this.currentRow) return this.svMessage({ type: 'warning', title: '请选中一条数据进行操作' })
  // 框架通用方法，打开新页签
  // name：router/index.js中配置的路由的name
  // title：标签名称
  // status：页签状态，自定义(当前文件中每个功能的status状态应该唯一)
  // id：当前行的id，$openTag方法会根据id生成路由，如果传的不是id是userId，需要传入pkName: 'userId'，userId: this.currentRow.id
  this.$openTag(this, {
    name: 'SvTableEditDemo',
    title: `用户查看-${this.currentRow.name}`,
    status: 'view',
    id: this.currentRow.id
  })
},
```

4.3 修改，传入this和参数，name：路由名称，title：标签页名称，status：当前状态edit，id：当前用户id（分别和router/index.js中path中配置的路由参数对应，同时也和子组件SvEditDemo中props接收的参数对应）

```
// 修改
handleEdit() {
  if (!this.currentRow) return this.svMessage({ type: 'warning', title: '请选中一条数据进行操作' })
  // 框架通用方法，打开新页签
  // name：router/index.js中配置的路由的name
  // title：标签名称
  // status：页签状态，自定义(当前文件中每个功能的status状态应该唯一)
  // id：当前行的id，$openTag方法会根据id生成路由，如果传的不是id是userId，需要传入pkName: 'userId', userId: this.currentRow.id
  this.$openTag(this, {
    name: 'SvTableEditDemo',
    title: `用户编辑-${this.currentRow.name}`,
    status: 'edit',
    id: this.currentRow.id
  })
},
```

5. 在子组件SvTableEditDemo中调用this.$closeTag方法关闭已打开的页签，传入this和cancelToPath，cancelToPath为关闭页签后要跳转的路由，非必填

```
data() {
  return {
    cancelToPath: '/demo/SvTableDemo',
```

```
// 提交数据
async handleSubmit() {
  // 新增
  if (this.dialogStatus === 'create') {
    this.$notify({
      title: '成功',
      message: '添加成功',
      type: 'success',
      duration: 2000
    })
    // 调用全局方法，关闭页签
    this.$closeTag(this, this.cancelToPath)
    // 修改
  } else {
    this.$notify({
      title: '成功',
      message: '修改成功',
      type: 'success',
      duration: 2000      You, 3 months ago • improvement: 去掉SvConfirm和SvTableContainer，重…
    })
    // 调用全局方法，关闭页签
    this.$closeTag(this, this.cancelToPath)
  }
 }
}
```