源码及演示环境地址

源码：http://192.168.0.221:7003/project-base/admin-demo-vue

演示环境：http://admin-demo.dev.fero.com.cn

项目生成器模板：http://192.168.0.221:7003/liusiyao/vue-template

更新日志

2.0

- 将 APP_KEY,PROJECT_NAME 配置移动到外层 config 中
- 将 css 样式文件夹内的样式文件重新拆分
  - base.css 基础样式重置
  - default.less element 样式覆盖、基础共通class样式
  - global.less 项目全局样式
  - theme.less 主题变量文件
  - utils.less less 可继承样式等通用方法
- src/assets/js/config.js 中 cookie 增加set
- http的修改
  - 增加 `409` 冲突状态处理，默认为刷新页面
  - 增加配置可绕过默认错误处理，完全自定义请求失败处理
  - 增加时间戳，防止请求缓存
- 将默认组件的样式从 default.less 移动到各自组件中
- 增加多个布局组件
- 增加 vue-meta 可以自定义网页标题

组件文档

authority-management 权限控制

用于根据授权功能列表进行页面元素的权限控制

> 依赖store中的featureList

基本用法

基础的组件使用方法

```
<template>
  <auth componentCode="features-code">
    <el-button type="primary">编辑</el-button>
  </auth>
</template>
<script>
  import auth from '@/components/authority-management'
  export default {
    components: {
      auth: auth
    },
  }
</script>
```

`componentCode` 的值是**预先定义**好的需要权限控制功能元素的功能code，该code不在功能列表中组件slot内容**不会**被渲染。

多code共通功能，支持简单逻辑判断

适用于同一个需要控制的元素具有多个功能code，如页面复用等情况

```
<template>
  <auth :componentCode="{code: ['export-credit-btn', 'approve-export-credit-btn'], condition: 'or'}">
    <el-button type="primary">导出PDF</el-button>
  </auth>
</template>
<script>
  import auth from '@/components/authority-management'
  export default {
    components: {
      auth: auth
    },
  }
</script>
```

`componentCode` 支持传递对象 `code` **必须**为数组； `condition` 为逻辑判断条件支持 `or` 和 `and`， `or` 只要数组中有一个code存在于功能列表中即有权限， `and` 为数组中的code在功能列表中都存在才有权限。

属性

参数	说明	类型	可选值	默认值
componentCode	功能code	string/object		
code	简写，与`componentCode`完全相同，两个都传只有`code`生效	string/object		

componentCode为对象时

参数	说明	类型	可选值	默认值
code	多个功能code	array		
condition	判断条件，不是`or`时按`and`处理	string	or/and	and

header 头部

固定在页面顶部的全局头部

基本用法

```
<template>
  <el-container direction="vertical">
    <page-header></page-header>
  </el-container>
</template>
<script>
  import pageHeader from '@/components/header'
  export default {
    components: {
      'page-header': pageHeader
    },
  }
</script>
```

> 暂不支持传参的方式定制化
>
> 副标题`projectTitle`来自`config/prod.env.js 中的 PROJECT_NAME`
>
> 用户名`empName`来自 `store`，点击logo返回的链接`consoleDomain`来自 `profiles.consoleDomain()`
>
> 无用的功能图标需要手动修改组件注释
>
> 退出需要 `api.base.logOut()`

page-title 标题栏

固定在内容区顶部的标题、操作栏

标题和操作都可通过slot自定义，返回按钮通过属性开启

标题默认通过路由中的`meta.title`获取当前页面标题，通过`meta.parentName`递归读取上级页面标题和路径，注意上级路径是使用动态路由(:id这种形式)的自动读取无法动态改变标题需要手动配置。

> 已集成在`src/layouts/default.vue`中

基本用法

```
<template>
  <el-main>
    <page-title :showBackBtn="true">
      <template slot="btn-inner">
        <el-button icon="iconfont icon-xinzeng" type="success">新增</el-button>
      </template>
    </page-title>
  </el-main>
</template>
<script>
  import pageHeader from '@/components/header'
  export default {
    components: {
      'page-header': pageHeader
    },
  }
</script>
```

标题自定义用法

```
<template>
  <el-main>
    <page-title :showBackBtn="true">
      <template slot="breadcrumb">
        <el-breadcrumb-item>页面名称</el-breadcrumb-item>
      </template>
      <template slot="btn-inner">
        <el-button icon="iconfont icon-xinzeng" type="success">新增</el-button>
      </template>
    </page-title>
  </el-main>
</template>
<script>
  import pageHeader from '@/components/header'
  export default {
    components: {
      'page-header': pageHeader
    },
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
show-back-btn	显示返回按钮	boolean	true/false	false

插槽

name	说明
breadcrumb	面包屑导航，请直接传入`<el-breadcrumb-item>`元素
btn-inner	按钮操作区

pagination 分页

分页组件，默认layout为`prev, pager, next, jumper`可根据需要直接修改组件

基本用法

```
<template>
  <pagination :total="total" :pageNum="pageNum" :pageSize="pageSize" :pages="pages"
:pageId="pageId" @changePageNum="changePageNum"></pagination>
</template>
<script>
  import pagination from '@/components/pagination'
  export default {
    components: {
      pagination: pagination
    },
    data () {
      return {
        total: 10,
        pageNum: 1,
        pageSize: 10,
        pages: 1,
        pageId: 'demo1'
      }
    },
    methods: {
      changePageNum ({pageId, pageNum}) {
        console.log({pageId, pageNum})
      }
    }
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
pageNum	页数	number		
total	总条数	number		
pageSize	每页条目数	number		
pages	总页数	number		
pageId	分页器ID	String/Number		

事件

事件名称	说明	回调参数
changePageNum	分页改变时触发	{pageId, pageNum}

search-inner 查询栏

放置查询条件表单的组件

基本用法

```
<template>
  <search-inner :searchId="demo1" :searchForm="searchForm" @submit-search="search" @clear-search="reset">
    <template slot-scope="scope">
      <el-row :gutter="20">
        <el-col :span="6">
          <el-form-item label="文字输入：">
            <el-input placeholder="请输入文字输入" v-model="searchForm.input1"></el-input>
          </el-form-item>
        </el-col>
      </el-row>
    </template>
  </search-inner>
</template>
<script>
  import searchInner from '@/components/search-inner'
  export default {
    components: {
      'page-header': pageHeader
    },
    data () {
      return {
        searchForm: {
          input1: ''
        }
      }
```

```
    },
    methods: {
      search ({searchId, searchForm}) {
      console.log(searchId, searchForm)
    },
    reset (searchId) {
      console.log('清空', searchId)
    },
    }
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
searchId	搜索框ID	number/string		
title	搜索框标题	String		条件筛选
searchBtn	搜索按钮文字，传`false`隐藏按钮	String/Boolean		搜索
clearBtn	清空按钮文字，传`false`隐藏按钮	String/Boolean		清空
labelPosition	表单域标签的位置	String		left
labelWidth	表单域标签的宽度	String		
searchForm	搜索条件	Object		{}

插槽

name	说明
-	请直接传入`<el-form-item>`元素，`searchForm`会通过`slot-scope`返回，在绝大多数情况用不上

事件

事件名称	说明	回调参数
search	点击搜索按钮触发	{searchId, searchForm}
reset	点击重置按钮触发	searchId

side 侧栏

固定在页面左侧的菜单栏

> 已集成在`src/layouts/default.vue`中

基本用法

```vue
<template>
  <el-container class="main-container">
      <page-side></page-side>
  </el-container>
</template>
<script>
  import pageSide from '@/components/side'
  export default {
    components: {
      'page-side': pageSide
    },
  }
</script>
```

table-inner 表格容器

整合标题、按钮、表格和分页的综合表格区域

基本用法

```vue
<template>
  <table-inner title="列表" :table-data="tableList.list" :pageNum="tableList.pageNum" :pageSize="tableList.pageSize" :pages="tableList.pages" pageId="test1" @changePageNum="changePageNum">
    <template slot="btn-inner">
      <el-button type="primary">按钮1</el-button>
    </template>
    <template slot="table-columns">
      <el-table-column prop="no" label="编号"></el-table-column>
    </template>
  </table-inner>
</template>
<script>
  import tableInner from '@/components/table-inner'
  export default {
    components: {
      'table-inner': tableInner
    },
    data () {
      return {
        tableList: {
          total: 10,
          pageNum: 1,
          pageSize: 10,
          pages: 1,
          list: [
            no: 'A001'
          ]
        }
      }
    },
```

```
    methods: {
      changePageNum ({pageId, pageNum}) {
        console.log({pageId, pageNum})
      }
    }
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
tableHeader	表格头部	Boolean	true/false	true
tableFooter	表格底部	Boolean	true/false	true
title	标题，位于头部内	String		搜索结果
border	边线	Boolean	true/false	false
tableData	表格数据	Array		[]
pageNum	页数	number		
total	总条数	number		
pageSize	每页条目数	number		
pages	总页数	number		
pageId	分页器ID	String/Number		

插槽

name	说明
btn-inner	标题右侧按钮区
table-bd	表格主体部分，用于完全自定义主体区域
table-columns	表格项，请直接传入`<el-table-column>`元素
table-ft	表格底部，用于完全自定义底部区域，传入后底部默认分页内容无效

事件

事件名称	说明	回调参数
changePageNum	分页改变时触发	{pageId, pageNum}

file-list 文件列表

用于放置附件的列表，有文字型和卡片型两种样式，可与图片预览组件结合使用

基础用法

```
<template>
  <default-layout>
    <!-- 其他无关元素已省略 -->
    <!-- 文字型 -->
    <file-list :files="fileList"></file-list>
    <!-- 图片型 -->
    <file-list :files="fileList" list-type="picture-card"></file-list>
  </default-layout>
</template>
<script>
  import fileList from '@/components/file-list'
  export default {
    components: {
      fileList
    },
    data () {
      return {
        fileList: [
          {
            name: '图片名称',
            src: 'http://xxx.com/file/img1.png'
          },
          {
            name: '文件名称',
            src: 'http://xxx.com/file/file.pdf'
          }
        ]
      }
    }
  }
</script>
```

自定义配置用法

```
<template>
  <default-layout>
    <!-- 其他无关元素已省略 -->
    <!-- 文字型 -->
    <file-list :files="fileList" :props="{name: 'name', src: 'url'}"></file-list>
    <!-- 图片型 -->
    <file-list :files="fileList" :props="{name: 'name', src: 'url'}" list-
type="picture-card"></file-list>
  </default-layout>
</template>
<script>
  import fileList from '@/components/file-list'
  export default {
    components: {
```

```
      fileList
    },
    data () {
      return {
        fileList: [
          {
            name: '图片名称',
            url: 'http://xxx.com/file/img1.png'
          },
          {
            name: '文件名称',
            url: 'http://xxx.com/file/file.pdf'
          }
        ]
      }
    }
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
listType	列表类型	String	text/picture-card	text
files	文件列表	Array		[]
nodeKey	节点ID	String		id
props	配置选项	Object		

props

参数	说明	类型	可选值	默认值
name	文件名	String		name
src	文件路径	String		src

事件

事件名称	说明	回调参数
handlePreview	点击时触发，传入会覆盖默认的预览行为	file

viewer 图片查看器

用来对图片进行预览的组件，支持图片组切换、放大缩小、旋转

基础用法

```
<template>
  <default-layout>
    <button @click="viewImg(file)">查看图片</p>
  </default-layout>
</template>
<script>
  import viewer from '@/components/viewer/index'
  export default {
    components: {
      fileList
    },
    data () {
      return {
        file: 'http://xxx.com/img/photo.jpg'
      }
    },
    methods: {
      viewImg (file) {
        viewer(file)
      }
    }
  }
</script>
```

带有文件信息

文件名称和路径必须为title和src

```
<template>
  <default-layout>
    <button @click="viewImg(file)">查看图片</p>
  </default-layout>
</template>
<script>
  import viewer from '@/components/viewer/index'
  export default {
    components: {
      fileList
    },
    data () {
      return {
        file: {
          src: 'http://xxx.com/img/photo.jpg‘,
          title: '图片标题'
        }
      }
    },
    methods: {
      viewImg (file) {
        viewer(file)
      }
    }
```

```
  }
</script>
```

多文件预览

```
<template>
  <default-layout>
    <template v-for="file in files">
      <button :key="file.id" @click="viewImg(file)">查看图片</p>
    </template>
  </default-layout>
</template>
<script>
  import viewer from '@/components/viewer/index'
  export default {
    components: {
      fileList
    },
    data () {
      return {
        files: [
          {
            id:1,
            src: 'http://xxx.com/img/photo1.jpg‘,
            title: '图片标题1'
          },
          {
            id:2,
            src: 'http://xxx.com/img/photo2.jpg‘,
            title: '图片标题2'
          }
        ]
      }
    },
    methods: {
      viewImg (file) {
        var fileList = []
        var imgIndex = 0
        this.files.forEach((item, index) => {
          if (item.id === file.id) {
            imgIndex = index
          }
          fileList.push({
            title: item.title,
            src: item.src
          })
        })
        viewer(fileList, {index: imgIndex})
      }
    }
  }
</script>
```

调用参数

参数	说明	类型	可选值	默认值
imgInfo	图片信息	String/Object		
options	配置	Object		

imgInfo

参数	说明	类型	可选值	默认值
title	图片名称	String		
src	图片链接	String		

options

参数	说明	类型	可选值	默认值
imgs	图片数组	Array		
index	预览图片在数组中的位置	Number		

节流按钮

用于需要防止双击的、重复提交的按钮，用法及参数与普通`el-button`没有区别，默认延时`300ms`，按钮点击立即触发，`300ms`内只执行一次，**注意节流按钮不能替代加载效果，加载效果也不能替代节流。**

基本用法

```
<template>
  <default-layout>
    <debounce-button icon="iconfont icon-tijiao" type="primary" @click="saveForm">节流按钮</debounce-button>
  </default-layout>
</template>
<script>
  import debounceButton from '@/components/debounce-button'
  export default {
    components: {
      debounceButton
    },
    data () {
      return {
      }
    },
    methods: {
      saveForm () {
        console.log(new Date())
```

```
      }
    }
  }
</script>
```

布局文档

default 默认布局

带有头部、侧栏、标题、主要内容区的默认页面布局，组件已在Vue初始化时注册，使用时无需引入，当然引入也没有问题

基本用法

```
<template>
  <default-layout>
    <template slot="breadcrumb">
      <el-breadcrumb-item>页面标题</el-breadcrumb-item>
    </template>
    <template slot="btn-inner">
      <el-button icon="iconfont icon-xinzeng" type="success">新增</el-button>
    </template>
    <!-- 其他页面元素 -->
  </default-layout>
</template>
<script>
  export default {
    components: {
    }
  }
</script>
```

属性

参数	说明	类型	可选值	默认值
show-back-btn	显示返回按钮	boolean	true/false	false
show-page-title	显示页面标题	boolean	true/false	true

插槽

name	说明
breadcrumb	面包屑导航，请直接传入`<el-breadcrumb-item>`元素
btn-inner	按钮操作区
-	默认插槽，用容纳主体区域的元素

simple 简单布局

一个简单的空白页布局，用于错误页、登录页等自定义页，组件已在Vue初始化时注册，使用时无需引入

基本用法

```
<template>
  <simple-layout>
    <!-- 任意页面元素 -->
  </default-layout>
</template>
<script>
  export default {
    components: {
    }
  }
</script>
```

插槽

name	说明
-	默认插槽，用于容纳页面元素

页面布局指引

参考在线演示环境（http://admin-demo.dev.fero.com.cn/）进行布局

列表页

- 对列表页展示的业务进行新增等针对业务本身的操作，放置在标题栏
- 搜索框，提供上下排列和左右排列两种样式。如果有搜索条件较长或要求输入框必须对齐等要求建议选择上下排列。搜索和重置按钮都提供关闭和自定义文字，但如果全局修改建议直接更改组件
- 搜索框每列4个，输入框较长的条件如时间范围占2个长度，一般不一个条件占一行
- 列表筛选数据的导出等针对有条件的数据，按钮放置在列表标题的右侧
- 列表按钮统一使用文字按钮
- 列表的左中右对齐根据项目实际要求设置
- tab内直接放置search-inner 和 table-inner 样式已进行调整无需额外更改

表单/详情页

- 默认情况下一行两列输入框，有文本域等情况可以一行一个，如有多行表单/详情与单行表单放在同一行，应注意对换行样式进行处理，避免样式错位
- 带tab的表单内可在表单上部或下部放置tab内的保存按钮，整体保存按钮应位于标题栏
- 注意仔细查看demo页面，表格容器与表单/详情容器是并列的
- 表单/详情的具体内容与标题有20px的缩进是为背景色预留的，在`.detail-inner`元素增加`.bg`即可开启
- 只有标题栏的按钮带图标，主要内容区的按钮都没有图标

- 表单/详情页的表格默认是带边线的
- 表单、表格、tabs组合边距已进行调整，一般无需额外更改
- 上传/文件展示，采用图片列表模式显示时，独占一行
- 新增按钮都使用绿色

内置功能/插件用法

vue-meta

页面标题会从路由中的`meta.title`读取，可被页面内的`metaInfo.title`覆盖，具体用法详见https://github.com/nuxt/vue-meta