组件项目地址: https://github.com/hejiyun/vue-element-model

1. 搜素表单

效果展示:

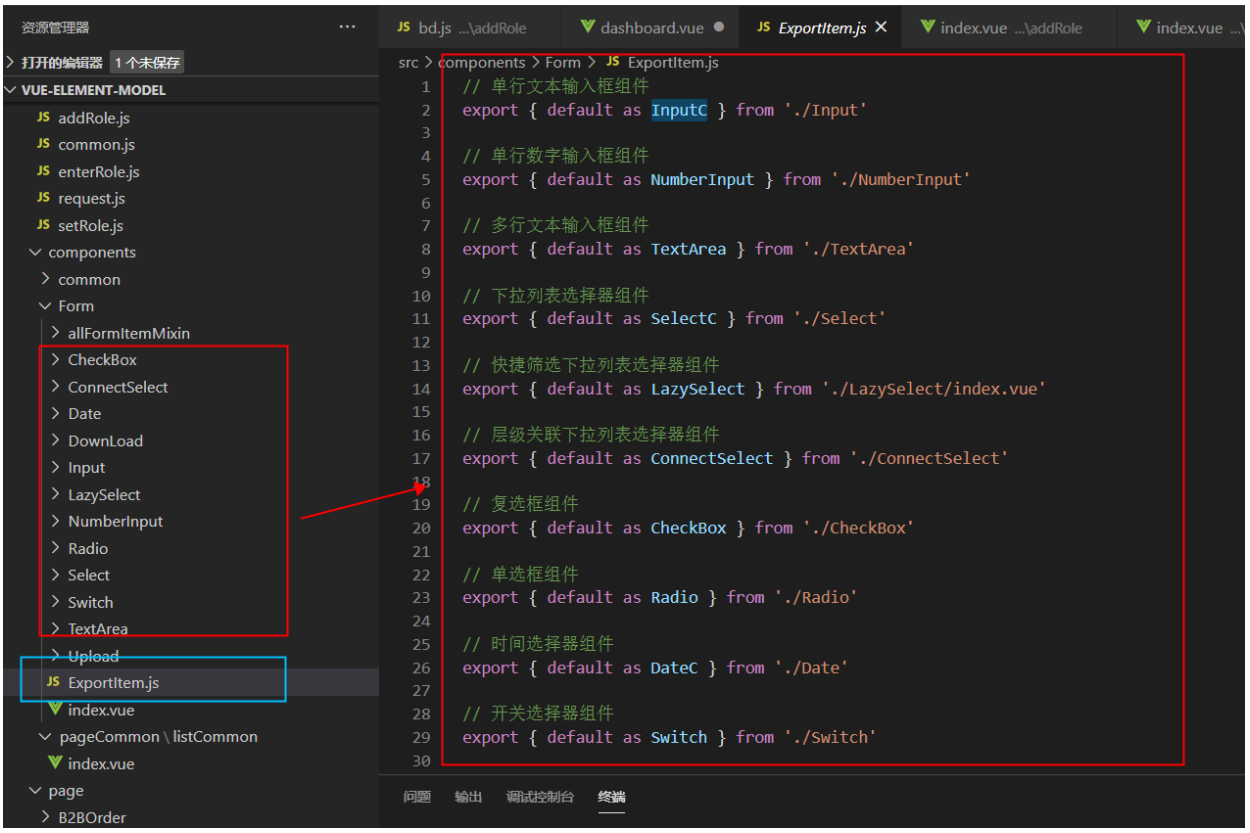
使用:

1. 引入对应搜素表单模块, 将form文件夹复制到项目中,

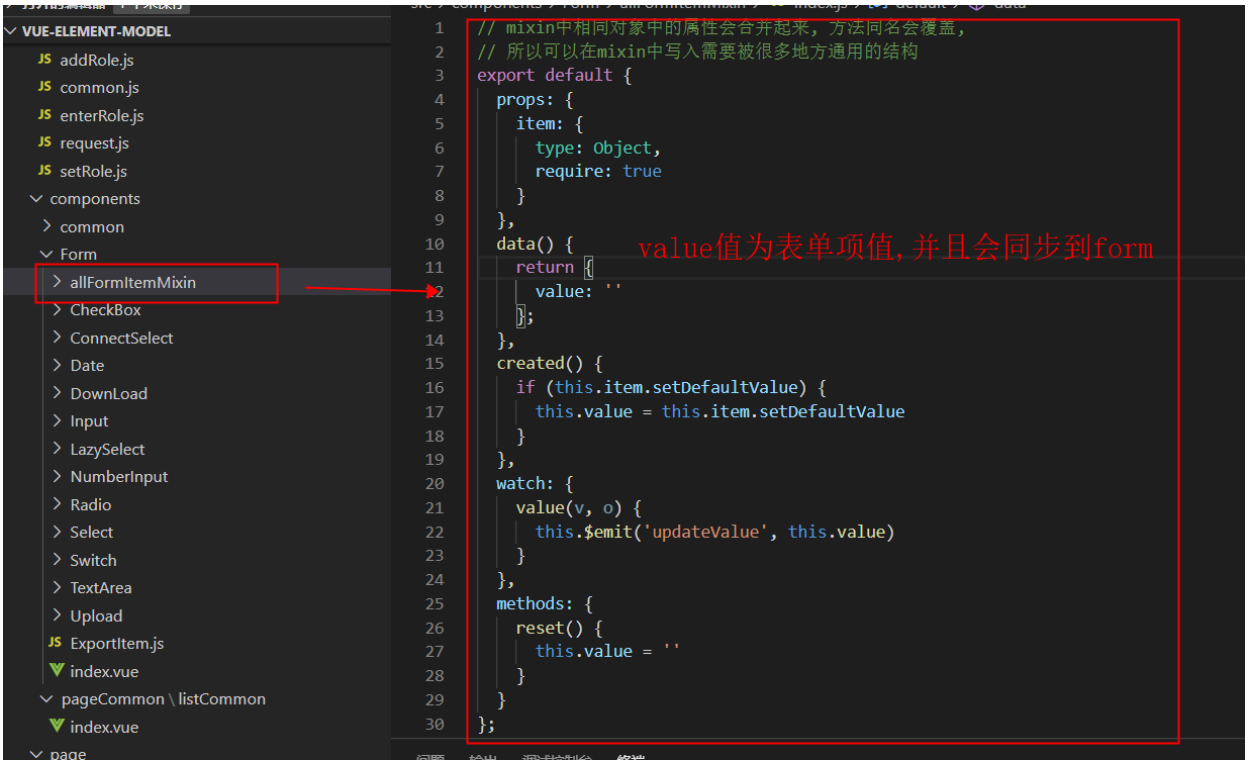
2. 然后在需要使用的组件中 import引入, 并注册, 使用标签, 其中options为该表单需要添加的表单项数组, submit为点击确认时, 触发的函数, 传递参数为当前form的值.

```vue
<div>
  <SearchForm v-if="!Config.editSearchBar" :options="Config.searchBar" @submit="submit">
    <template v-for="item in Config.searchBar" slot-scope="scope" :slot="item.operate ? item.prop : null">
      <slot v-if="item.operate" :params="scope.params" :name="`search-${item.prop}`"/>
    </template>
    <template v-slot:btn="params">
      <slot :params="params" name="search-btn"/>
    </template>
  </SearchForm>
  <slot v-else name="searchBar"/>
  <TableCommon v-loading="loading" :table-cofig="Config.tableCofig" class="margin-top-15">
    <template v-for="item in Config.tableCofig.tableHeader" slot-scope="scope" :slot="item.operate ? item.prop : null">
      <slot v-if="item.operate" :row="scope.row" :name="`table-${item.prop}`"/>
    </template>
  </TableCommon>
  <Pagination ref="listCommonPage" :page-config="Config.pageConfig" @query="query" @loading="load"/>
</div>
</template>
<script>
import SearchForm from '@component/Form'
import TableCommon from 'common/Table'
import Pagination from 'common/pagination'
export default {
  name: 'ListPageCommon',
  components: {
    SearchForm,
    TableCommon,
    Pagination
  },
```

3. 表单组成部分, 该封装表单便捷之处在于可以自定义表单项, 自己想创建什么就可以照理创建, 并且在创建完成后, 需要在exportItem.js中导出,



4. 其中, allFormItemMixin文件夹为表单项通用配置提取, 表单项使用以注入mixin方式.



表单项注入mixin. 需要自定义可以依据mixin特性自己添加

5. 使用数据创建表单项, 上述标签上的options属性接收的是数组形式的数据, 表单会通过options的数组数据来创建对应个数的表单项

下面是各表单项使用截图. 其中 label 和prop 及cmp, rules, disabled为通配属性, 分别对应, 表单项标题/ 表单项对应form属性 / 表单项对应使用的组件 / 校验规则 / 是否禁用, mapping为数据源取值映射, 自动识别字典数据value,label 属性: 在options数组中, 具有下拉性质的配置中, 设置mapping属性, 属性值为数组,第一个为value对应字段, 第二个为label对应字段

```
np-b2b-cms > internal > src > basic_database > module_ods > JS delivery_order.js > config > searchBar
1  import { InputC, SelectC, HeadSelect, DateC, LazySelect, CheckBox } from '@component/Form/ExportItem.js'
2  import { getTableList } from '@/axios/module_ods/delivery_order'
3  import { getCurrentTime } from '@/utils/globalFunction'
4
5  const arr = getCurrentTime()
6
7  const config = {
8    searchBar: [
9      {
10       cmp: InputC,
11       prop: 'billNo',
12       placeholder: '请输入查询工单号',
13       label: '发货单号'
14     },
15     {
16       cmp: SelectC,
17       prop: 'billStatusList',
18       options: [],
19       multiple: true,
20       placeholder: '全部',
21       mapping: ['dictCode', 'dictValue'],
22       label: '发货单状态'
23     },
24     {
25       cmp: LazySelect,
26       prop: 'fromExternalWarehouseCodeList',
27       AllList: [],
28       optionHeader: ['仓库代码/externalWarehouseCode', '仓库名称/warehouseName', '仓库类型/warehouseTypeName'],
29       placeholder: '请选择发货仓',
30       label: '发货仓'
31     },
32     {
33       cmp: LazySelect,
34       prop: 'toExternalWarehouseCodeList',
35       AllList: [],
36       optionHeader: ['仓库代码/externalWarehouseCode', '仓库名称/warehouseName', '仓库类型/warehouseTypeName'],
37       placeholder: '请选择目的仓',
```

数据源的value和label字段自动识别映射

数据源

前一个为标题，后一个为对应取值


```
{
  cmp: HeadSelect,
  prop: 'vendCustCode',
  options: [],
  filterable: true,
  placeholder: '请选择客商',
  optionHeader: ['客商代码', '客商名称'],
  mapping: ['vendCustCode', 'vendCustName'],
  label: '客商名称'
},
{
  cmp: SelectC,
  prop: 'bizTypeList',
  options: [],
  multiple: true,
  placeholder: '全部',
  mapping: ['dictCode', 'dictValue'],
  label: '业务类型'
},
{
  cmp: InputC,
  prop: 'orderNo',
  placeholder: '请输入订单号',
  label: '订单号'
},
{
  cmp: InputC,
  prop: 'sourceBillNo',
  placeholder: '请输入来源单号',
  label: '来源单号'
},
{
  cmp: InputC,
  prop: 'shopBillNo',
  placeholder: '请输入网店交易单号',
```

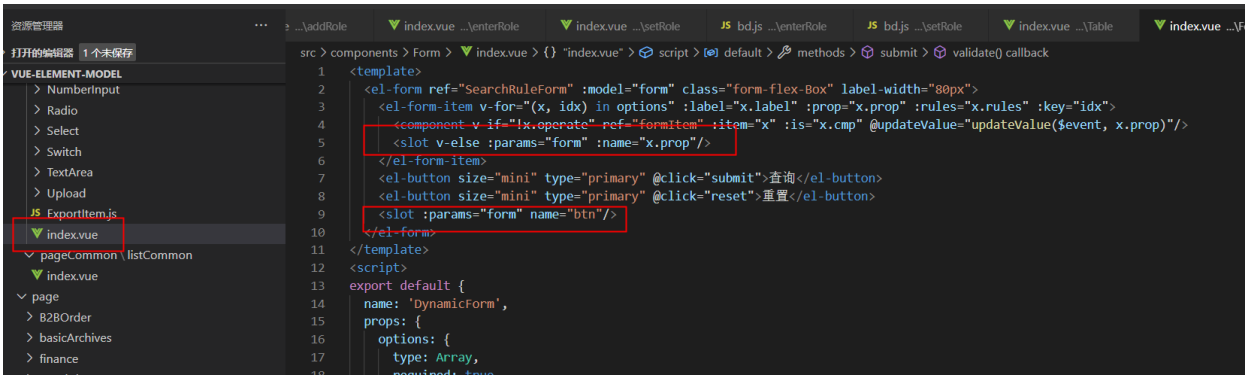
```
    {
      cmp: InputC,
      prop: 'phone',
      placeholder: '请输入收件人电话',
      label: '收件人电话'
    },
    {
      cmp: InputC,
      prop: 'expressBillNo',
      placeholder: '请输入快递单号',
      label: '快递单号'
    },
    {
      cmp: CheckBox,
      prop: 'hasDiffExpectQty',
      placeholder: '预计发货与实际发货有差异',
      label: ''
    },
    {
      cmp: DateC,
      prop: 'createTime',
      label: '发货单生成时间',
      splitKey: ['fromCreateTime', 'toCreateTime'],
      setDefaultValue: arr
    },
    {
      cmp: DateC,
      prop: 'deliveryTime',
      label: '发货时间',
      splitKey: ['fromDeliveryTime', 'toDeliveryTime']
    }
  ],
  tableCofig: {
```

时间控件自动解析参数，数组属性为需要解析的字段属性名

6. 自定义插入, 和自定义按钮: 在组件标签中, 使用template插槽可以对应插入自定义内容

插入按钮时, 使用 name="btn", 插入某一项自定义内容时, name使用当前项prop属性值, 并且需要

在options数组当前项中添加operate:true属性, 作为插入属性



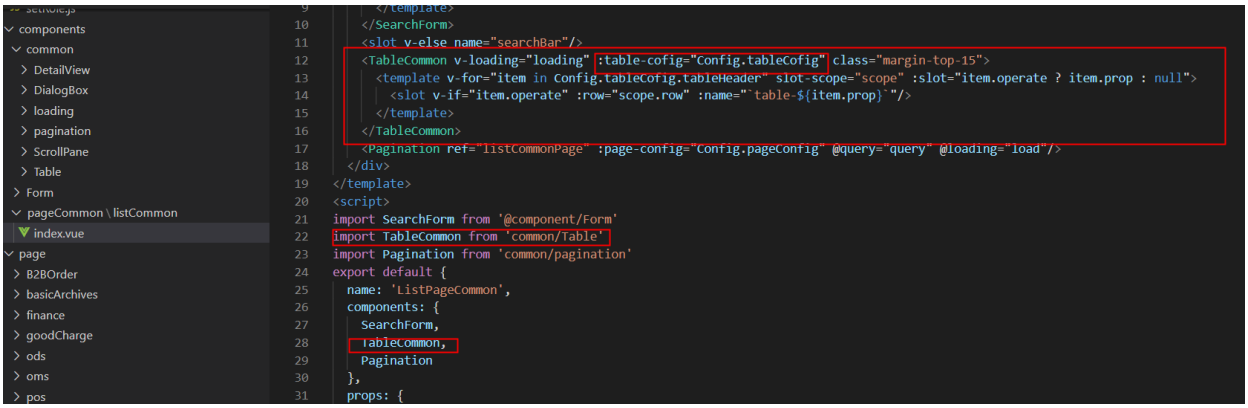
2. 表格

效果展示:

序号	用户名	角色	创建时间	操作
1	张岩	管理员	2019-03-01T08:12:40.000+0000	编辑
2	姚宇元	资讯	2019-02-28T09:26:59.000+0000	编辑
3	管理员		2019-02-27T09:52:19.000+0000	编辑
4	刘蕾 Lenny	商品,财务,运营规划,项目,供应链,UED	2019-02-27T09:52:19.000+0000	编辑
6	吴静华	管理员,超级管理员	2019-02-27T09:52:19.000+0000	编辑
7	王娇	采购	2019-02-27T09:52:19.000+0000	编辑
8	杨俊	管理员	2019-04-01T06:50:47.000+0000	编辑
9	张敬宇	商品,运营,供应链	2019-04-03T06:13:00.000+0000	编辑

1.引入对应table模块, 将table文件夹复制到项目中

2. 在需要的组件中, 引入, 注册, 并使用标签, 其中tableCofig为表格配置,



3. tableCofig默认属性, 多选方法,及排序方法, 都是$emit传递参数, 参数为当前列/ 当前选中

```
  },
  computed: {
    Config() {
      return Object.assign({
        tableHeader: [], // 表头
        expand: false, // expand 同element
        mergeTable: '', // 合并表格，传递合并属性字段，应为数组类型
        mergeTableColumnIndex: [], // 合并列index
        tableData: [], // 表格数据源
        checkFlag: false, // 多选
        height: 0, // 表格 基于视图差高度  :height="`calc(100vh - ${Config.height}px)`"
        changeCell: [], // 改变列样式
        changeCellColor: 'red', // 默认红色
        changeCellType: 'color' // 默认改变列颜色
      }, this.tableCofig)
    }
  },
  watch: {
```

```
:span-method="Config.mergeTable && Config.mergeTableColumnIndex.length ? arraySpanMe
border
@sort-change="getSortList"
@selection-change="handleSelectionChange">
<el-table-column v-if="Config.expand" type="expand">
  <template slot-scope="scope">
    <slot :row="scope.row" name="expand"/>
  </template>
</el-table-column>
<el-table-column
  v-if="Config.checkFlag"
  type="selection"
  width="55"/>
```

排序

多选

4. 表格赋值简单数据示例：operate属性依然作为插槽属性，可自定义内容

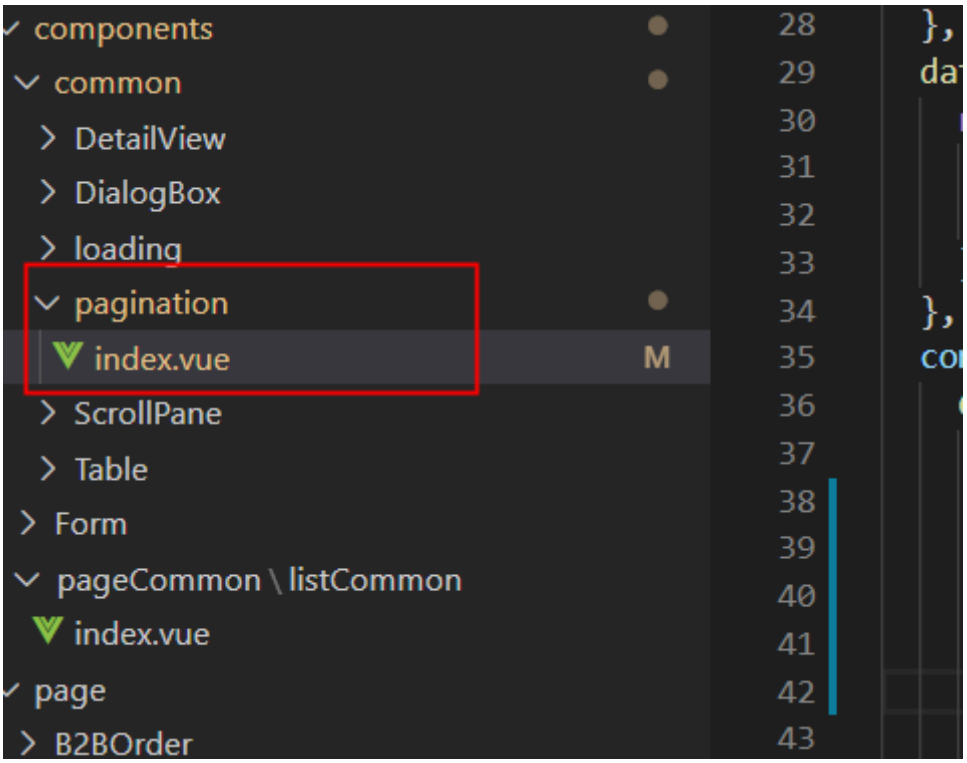
```
tableCofig: {
  height: 200,
  tableHeader: [
    {
      label: '序号',
      prop: 'id'
    },
    {
      label: '用户名',
      prop: 'userName'
    },
    {
      label: '角色',
      prop: 'roleNames',
      operate: true
    },
    {
      label: '创建时间',
      prop: 'createTime'
    },
    {
      label: '操作',
      prop: 'btn',
      operate: true
    }
  ],
  tableData: []
},
```

5. 合并表格思路, 对于合并表格, 赋值mergeTable之后, table内部会以mergeTable对应的属性值(数组)遍历,
然后按数组内子项个数进行拆分, 并合并父属性, 形成新的表格(比原数据多行,) 然后会记录子项有多少个,
基于这个个数, 去对拆分后的表格进行合并项, 合并完成后, 表格行依然是原来行数.

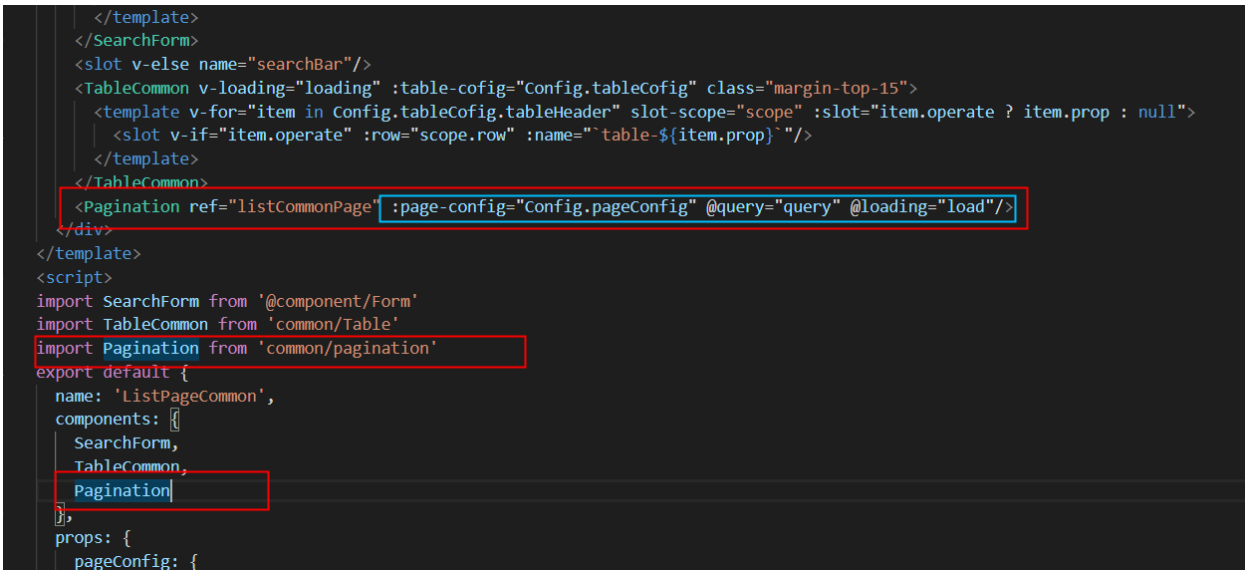
3. 分页器

效果展示:

1. 引入对应pagination模块, 将pagination文件夹复制到项目中



2. 然后在需要使用的组件中 import引入, 并注册, 使用标签, 其中pageConfig为该分页器的配置集合, query为接口返回数据列表回调方法, 参数为返回数据. loading为loading动画回调方法, 如果需要添加loading动画, 则该方法参数为BOOLEAN值, 返回参数.



3.pageCofig默认配置, 其中baseList可以设置接口传递中的num 和size对应的字段, num对应
baseList中第一个项, size对应第二个

```
    }
  },
  computed: {
    Config() {
      return Object.assign({
        pageSizeList: [10, 20, 50, 100], // 每页多少条
        layout: 'total, sizes, prev, pager, next, jumper', // 分页器格式，这里默认全部，可以自己修改
        baseList: [], // 传递，page，limit取值数组，定义传递参数属性名，第一个参数为当前页，第二个参数为条数
        noGetList: false, // 第一次是否发送请求
        request: function() { // 请求数据的方法请求
          return
        }
      }, this.pageConfig)
    }
  }
```

4. 简单示例：添加属性 `noGetList: true,` 可以取消第一次初始化请求.

```
      tableData: []
    },
    pageConfig: {
      baseList: ['page', 'limit'],
      request: getTableList // 请求方法
    }
  }

  const DialogConfig = {
    options: [
```

```
    getList(params) {
      this.$refs.listCommonPage.getList(params)
    },
    query(data) {
      this.Config.tableCofig.tableData = data
    },
    load(B) {
      this.loading = B
    }
  }
}
```

5. pagination组件本身具有geList方法, 可传递形参params,为对象类型数据. 调用该方法可重新请求接口数据

4. 详情展示

效果展示:

发货单号	9725649849221158-D001			订单号	SO1999725649849221158		
来源单号				网店交易单号			
发货单状态	已完成	发货单生成时间	2020-09-15 11:15:00	发货单下发时间	2020-09-15 11:16:01	发货时间	2019-08-29 15:00:31
业务类型	B2B发货	客商编码	11000050	客商名称	上海玉柱大宗		
调拨类型		调拨方式名称		调拨优先级			
发运方式	其他约定的付款方式	所属批次		预计到货时间		其他单号	
发货仓代码	WH001233	发货仓名称	宝唯东莞-代采买期货大宗销售仓	目的仓代码		目的仓名称	
收件人省市区	江苏省 连云港市 灌南县	收件人地址	连云港市浦南镇许安路与淘宝路路口八...	收件人姓名	八号仓	收件人电话	18136575003
是否打印发票	否	是否货票同行	否			备注	

1. 引入详情展示组件, 将detailView文件夹复制到项目中

2. 在需要使用的组件中, 引入, 注册, 使用标签，标签接收两个属性options与data，data为需要展示的数据源
对象，options为数组形式数据, 设置各展示项的基本配置

```
<div class="padding-left-10">
  <detail-view :options="detailOptions" :data="baseText" />
</div>
```

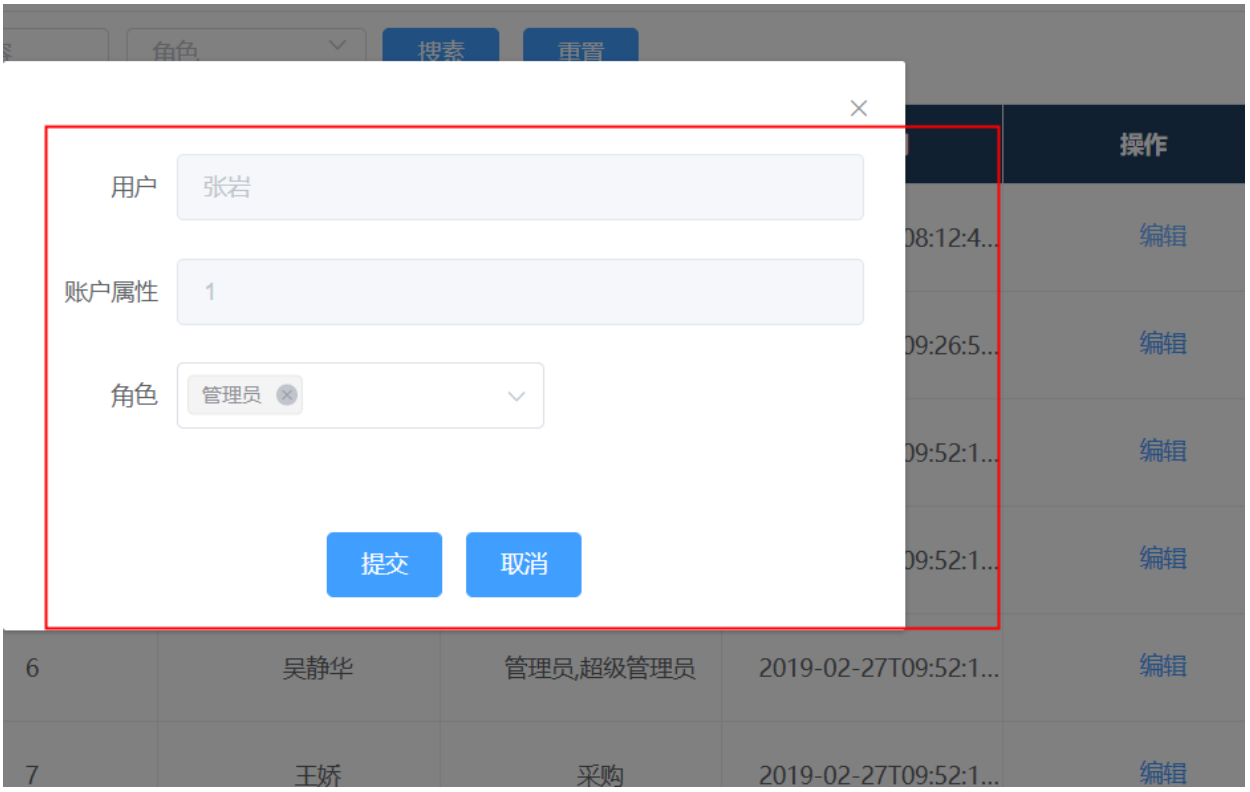
```
</template>

<script>
import PageCommon from '@/components/PageCommon'
import TableCommon from '@/components/TableCommon'
import detailView from '@/components/detailView'
import TableText from '@/textResources/tableText'
import detailText from '@/textResources/detail'
// import moment from 'moment'
export default {
  name: 'DeliveryDetail',
  components: {
    PageCommon, // 分页器组件
    TableCommon, // 表格组件
    detailView // 详情组件
  },
  data() {
    return {
```

3. 简单示例: data为返回详情信息对象, options设定每一项, label为展示名, key为对应data中取值属性, operate可自定义插槽

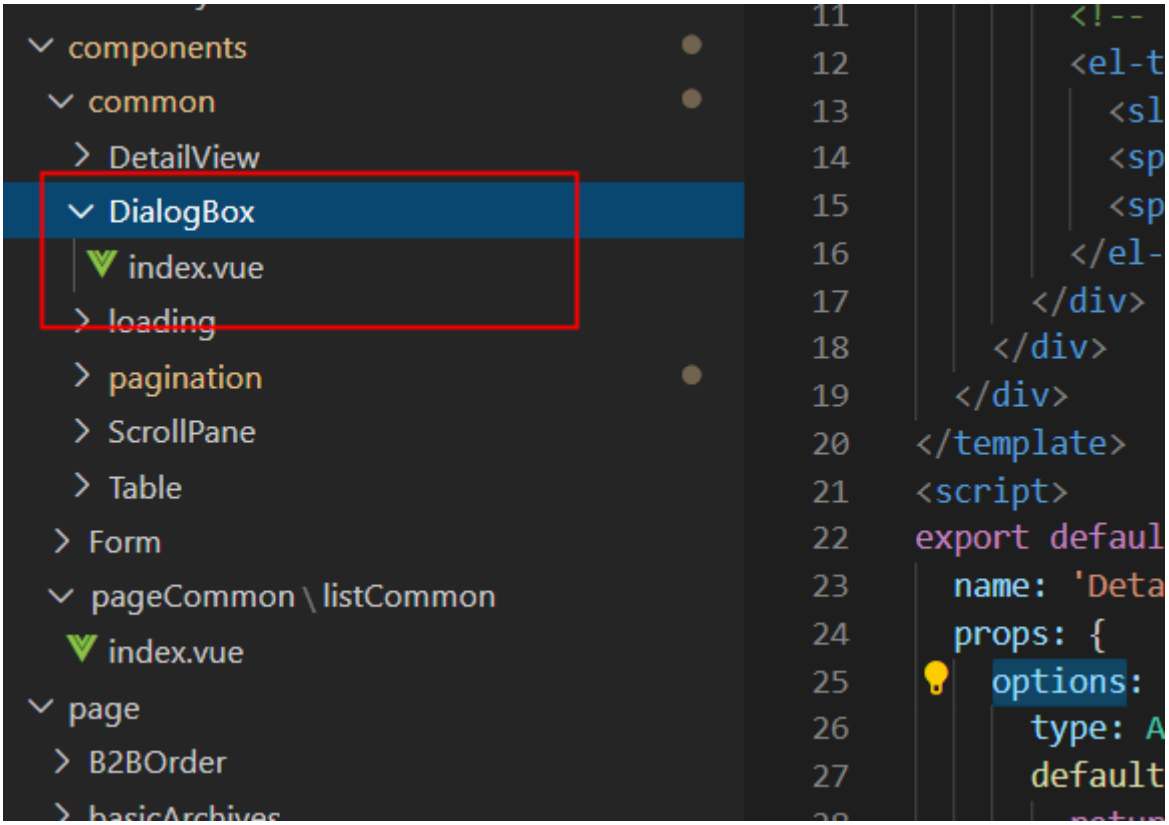
```
OMSOrderDetailOptions: [{
  label: '交易单号',
  key: 'orderNo'
}, {
  label: '交易订单创建时间',
  key: 'orderTransactTime'
}, {
  label: '平台单号',
  key: 'platformNo'
},
{
  label: '平台订单生成时间',
  key: 'orderCreateTime'
}, {
  label: '外部单号',
  key: 'externalNo'
}, {
  label: '单据状态',
  key: 'orderStatusName'
}, {
  label: '单据类型',
  key: 'orderTypeName'
}, {
  label: '业务类型',
  key: 'bizTypeName'
}, {
  label: '客店代码',
  key: 'shopCode'
```

5. 弹窗

效果展示:



1. 引入弹窗组件, 将DialogBox复制到项目中



2. 在需要使用的组件中, 引入, 注册 并使用标签, showDialog控制弹窗显示隐藏, 使用.sync

.

dialogCofig配置弹窗内内容, 这里以表单为主, 看表单就可以了,

DilalogConfirm 是点击提交的回调函数, 参数为弹窗内form的值对象.

```
        <span>{{ scope.row.roleNames.join(',') || scope.row.roleNames[0] }}</span>
      </template>
    </HtmlPage>
    <DialogBox
      :dialog-config="DialogConfig"
      :show-dialog.sync="showDialog"
      @DialogConfirm="edit"/>
  </div>
</template>
<script>
import SearchBar from './searchBar'
import HtmlPage from '@component/pageCommon/listCommon'
import { config, DialogConfig } from './bd'
import { getRoleList } from '@/axios/enterRole';
import DialogBox from 'common/DialogBox'
export default {
  name: 'EnterRole',
  components: {
    HtmlPage,
    SearchBar,
    DialogBox
  },
  data() {
```

3. dialogConfig配置, 且组件本身提供reset方法, 重置内部表单(关闭时会自动触发)

```
49    },
50    computed: {
51      Config() {
52        return Object.assign({
53          options: [],
54          cancelText: '取消',
55          confirmText: '提交',
56          widthPercent: 30, // 弹窗宽度
57          dialogText: '', // 弹窗标题
58          hasFooter: true // 是否有底部按钮
59        }, this.DialogConfig)
60      }
61    },
62    watch: {
63      showDialog(val) {
64        this.dialogVisible = val
```

4.自定义按钮和自定义子项, 使用插槽插入, 按钮插入对应 name="btn" ， 子项使用operate插入

```
    center
    @open="open"
    @close="handleClose">
    <el-form ref="dialogBoxForm" :model="form" class="dialog-bar" label-width="80px">
      <el-form-item v-for="(x, idx) in Config.options" :label="x.label" :prop="x.prop" :rules="x.rules" :key="idx">
        <component v-if="!x.operate" ref="dialogFormItem" :item="x" :is="x.cmp" @updateValue="updateValue($event, x.prop)"/>
        <slot v-else :params="x" :name="x.prop"/>
      </el-form-item>
    </el-form>
    <span v-if="Config.hasFooter" slot="footer" class="dialog-footer">
      <el-button v-if="Config.confirmText" type="primary" @click="DialogConfirm">{{ Config.confirmText }}</el-button>
      <el-button v-if="Config.cancelText" type="primary" @click="reset">{{ Config.cancelText }}</el-button>
      <slot :params="form" name="btn"/>
    </span>
  </el-dialog>
</div>
/template>
```

5. 简单示例

```
const addRoleConfig = {
  options: [
    {
      label: '角色名称',
      prop: 'roleName',
      cmp: InputC,
      rules: {
        required: true, message: '请输入角色名称', trigger: 'blur'
      }
    },
    {
      label: '角色英文名',
      prop: 'roleCode',
      cmp: InputC,
      rules: {
        required: true, message: '请输入角色英文名', trigger: 'blur'
      }
    },
    {
      label: '角色描述',
      prop: 'remark',
      cmp: TextArea,
      rules: {
        required: true, message: '请填写角色描述', trigger: 'blur'
      }
    }
  ]
```

6. 列表页

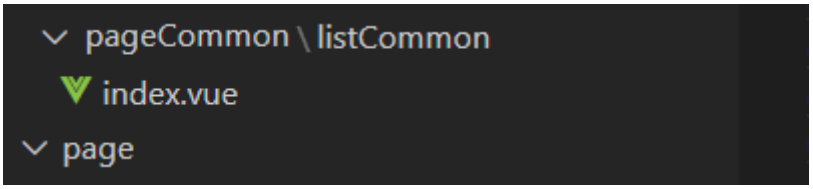
效果展示:

请输入内容 角色 搜索 重置

序号	用户名	角色	创建时间	操作
1	张岩	管理员	2019-03-01T08:12:4...	编辑
2	姚宁元	资讯	2019-02-28T09:26:5...	编辑
3	管理员		2019-02-27T09:52:1...	编辑
4	刘蕾 Lenny	商品,财务,运营规划,项...	2019-02-27T09:52:1...	编辑
6	吴静华	管理员,超级管理员	2019-02-27T09:52:1...	编辑
7	王娇	采购	2019-02-27T09:52:1...	编辑
8	杨俊	管理员	2019-04-01T06:50:4...	编辑
9	张敬宇	商品,运营,供应链	2019-04-03T06:13:0...	编辑

共 346 条 10条/页 1 2 3 4 5 6 ··· 35 前往 1 页

1. 引入列表页组件, 将pageCommon文件夹复制到项目中



2. 在需要的组件中引入, 注册, 使用标签

```vue
<div class="set-role">
  <HtmlPage ref="targetPage" :page-config="config">
    <template v-slot:searchBar>
      <el-button type="primary" @click="addDialog = true">新建角色</el-button>
    </template>
    <template v-slot:table-btn="scope">
      <el-button type="text" @click="view(scope.row)">查询</el-button>
      <el-button type="text" @click="showDialog = true">设置权限</el-button>
      <el-button type="text" @click="del(scope.row)">删除</el-button>
    </template>
  </HtmlPage>
  <DialogBox
    :dialog-config="addRoleConfig"
    :show-dialog.sync="addDialog"
    @DialogConfirm="add"/>
  <DialogBox
    :dialog-config="DialogConfig"
    :show-dialog.sync="showDialog"
    @DialogConfirm="editRole"/>
</div>
</template>
<script>
import HtmlPage from '@component/pageCommon/listCommon'
import { config, DialogConfig, addRoleConfig } from './bd'
// import { getRoleList } from '@/axios/setRole';
import DialogBox from 'common/DialogBox'
export default {
  name: 'SetRole',
  components: {
    HtmlPage,
```

3. pageCofig配置, 实质就是搜素/ 表格/分页的组合，需要注意的就是嵌套插槽的问题

```vue
  }
},
computed: {
  Config() {
    return Object.assign({
      editSearchBar: false, // 控制是否自定义编辑搜素模块，自定义则searchBar不会渲染
      searchBar: [], // 搜素模块
      tableCofig: {}, // 表格模块
      pageConfig: {} // 分页器模块
    }, this.pageConfig)
  }
},
methods: {
  submit(params) {
```

4. 简单示例

```vue
<template>
  <div>
    <SearchForm v-if="!Config.editSearchBar" :options="Config.searchBar" @submit="submit">
      <template v-for="item in Config.searchBar" slot-scope="scope" :slot="item.operate ? item.prop : null">
        <slot v-if="item.operate" :params="scope.params" :name="`search-${item.prop}`"/>
      </template>
      <template v-slot:btn="params">
        <slot :params="params" name="search-btn"/>
      </template>
    </SearchForm>
    <slot v-else name="searchBar"/>
    <TableCommon v-loading="loading" :table-cofig="Config.tableCofig" class="margin-top-15">
      <template v-for="item in Config.tableCofig.tableHeader" slot-scope="scope" :slot="item.operate ? item.prop : null">
        <slot v-if="item.operate" :row="scope.row" :name="`table-${item.prop}`"/>
      </template>
    </TableCommon>
    <Pagination ref="listCommonPage" :page-config="Config.pageConfig" @query="query" @loading="load"/>
  </div>
</template>
```

```
const config = {
  editSearchBar: true,
  tableCofig: {
    height: 200,
    tableHeader: [
      {
        label: '序号',
        prop: 'id'
      },
      {
        label: '用户名',
        prop: 'userName'
      },
      {
        label: '角色',
        prop: 'roleNames',
        operate: true
      },
      {
        label: '创建时间',
        prop: 'createTime'
      },
      {
        label: '操作',
        prop: 'btn',
        operate: true
      }
    ],
    tableData: []
  },
  pageConfig: {
    baseList: ['page', 'limit'],
    request: getTableList // 请求方法
  }
}
```