# 基本使用

<v-table

:*table-btns-config*="tableBtnsConfig"

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*label-width*="130"

:*get-data*="list"

@*gotoDict*="(rowData)=>{gotoPage(rowData, 'Dict')}"

@*gotoBusiness*="(rowData)=>gotoPage(rowData,'Business')"

@*gotoUser*="(rowData)=>gotoPage(rowData,'User')" />

tableBtnsConfig: [

{

name: '编辑',

editConfig: {

title: '编辑对接系统',

handler: update,

},

},

{

name: '字典维护',

eventName: 'gotoDict',

},

{

name: '对接用户管理',

eventName: 'gotoUser',

},

{

name: '对接商家管理',

eventName: 'gotoBusiness',

},

],

topBtnsConfig: [

{

name: '添加',

btnType: { isPlain: true, type: 'primary' },

addConfig: {

title: '添加对接系统',

handler: add,

},

},

],

columns: [

{

name: '对接系统',

id: 'name',

required: true,

support: ['query', 'add', 'edit'],

},

{

name: '对接商家数',

id: 'supplierCount',

},

{

name: '接入状态',

id: 'state',

required: true,

queryType: 'select',

options: [{ name: '启用', id: 0 }, { name: '禁用', id: *-*1 }],

support: { query: {}, edit: { type: 'radio' }, add: { type: 'radio' } },

},

{

name: '最近更新人',

id: 'updateUser',

},

{

name: '最近更新时间',

id: 'updateTime',

},

],

# 字段需要格式化显示，例如时间，例如状态

## 使用

* 前端定义选项，传入options字段或者直接传入formatter函数

columns: [

{

name: '状态',

id: 'state',

queryType: 'select',

required: true,

options: [{ name: '启用', id: 0 },

{ name: '禁用', id: *-*1 }],

support: {

add: { type: 'radio' },

query: {},

},

},

],

* 从后端选项，传入source函数和sourceFormat选项对应关系

columns: [

{

name: '区域',

required: true,

id: 'areaId',

queryType: 'select',

support: ['add', 'edit'],

source: findRegion,

sourceFormat: {

value: 'regionID',

label: 'regionName',

},

},

],

## 实现

有formatter优先用formatter，否则根据options生成

vTable.vue

mounted() {

this.initCount *=* this.columns.filter(one => one.source).length;

this.columns.forEach(column => {

*// 初始化options*

*if* (column.source) {

column.source().then(({ model }) => {

*if* (*!*column.formatter) {

this.generateFormatter(column);

}

this.currentCount*++*;

this.$set(column, 'options', model); *//*

}).catch(e => {

this.$message.error(`获取${column.name}选项失败!`);

console.log(e);

});

} *else* *if* ((column.options *&&* column.options.length) *||* ['updateTime'].includes(column.id)) {

*if* (*!*column.formatter) {

this.generateFormatter(column);

}

}

...

});

},

从options中匹配数据，匹配不到显示后端返回原始值

generateFormatter(column) {

column.formatter *=* val => {

*// 统一格式化时间*

*if* (column.id *===* 'updateTime') {

*return* formatDate(val);

}

const target *=* column.options.find(one => one[column.sourceFormat

*?* column.sourceFormat.value

*:* 'id'] *==* val);

*return* target *?* target[

column.sourceFormat

*?* column.sourceFormat.label

*:* 'name'

] *:* val;

};

},

# 某些字段需要查询

## 使用：support传入数组，数组包含query或者传入对象包含query属性

docList.vue

columns: [

{

name: '文档名称',

id: 'name',

support: ['query'],

},

]

Business.vue

columns: [

{

name: '供应商ID',

required: true,

id: 'supplierId',

support: {

query: {},

add: {

eventName: 'addIdChange',

},

edit: {

eventName: 'editIdChange',

disabled: true,

},

},

},

]

## 实现：

1.初始化查询表单

mounted() {

...

this.columns.forEach(column => {

*// 初始化options*

...

*// 初始化查询表单*

*if* (column.support) {

*// 是数组时，是最基础的配置，复杂的配置时是一个对象*

*if* (Array.isArray(column.support)) {

*if* (column.support.includes('query')) {

this.$set(this.queryParams, column.id, '');

}

} *else* *if* (*typeof* column.support *==* 'object') {

*if* (column.support.query) {

this.$set(this.queryParams, column.id, '');

}

...

}

}

});

},

2.筛选出显示的支持查询的字段

computed: {

showQueryColumns() {

*return* this.columns.filter(one => {

*if* (Array.isArray(one.support)) {

*return* one.support.includes('query');

} *if* (*typeof* one.support *==* 'object') {

*return* *!!*one.support.query;

}

*return* false;

});

},

}

# 某些字段支持新增或者编辑

## 使用：

Columns中需要编辑或者新增的字段传入edit 或 add 配置，

支持添加时，vTable传入topBtnsConfig，以配置弹框按钮和保存触发的函数

topBtnsConfig: [

{

name: '添加',

btnType: { isPlain: true, type: 'primary' },

addConfig: {

title: '添加对接供应商',

handler: add,

},

},

],

columns: [

{

name: '区域',

required: true,

id: 'areaId',

queryType: 'select',

support: ['add', 'edit'],

source: findRegion,

sourceFormat: {

value: 'regionID',

label: 'regionName',

},

},

]

## 实现：

handleTableBtnClick(config, rowData) {

*if* (config.editConfig) {

const index *=* this.tableDataHandled.indexOf(rowData);

this.addOrEdit(config.editConfig.title, rowData, index);

}

...

},

handleTopBtnClick(config) {

*if* (config.addConfig) {

this.addOrEdit(config.addConfig.title);

}

...

},

addOrEdit(title, rowData, index) {

let mode *=* 'add';

*if* (rowData) {

mode *=* 'edit';

this.dataForDialog.index *=* index;

this.dataForDialog.form *=* rowData;

}

this.dataForDialog.mode *=* mode;

this.dataForDialog.items *=* this.columns.filter(one => one.support *&&* ((Array.isArray(one.support) *&&* one.support.includes(mode)) *||* one.support[mode]));

this.dataForDialog.title *=* title;

this.$emit('beforeDialogOpen', this.dataForDialog, mode);

this.dataForDialog.show *=* true;

},

# 表格的查询和编辑弹组件类型不一致

## 使用：传入type，否则直接拿queryType作为弹框组件

System.vue

columns: [

{

name: '接入状态',

id: 'state',

required: true,

queryType: 'select',

options: [{ name: '启用', id: 0 },

{ name: '禁用', id: *-*1 }],

support: { query: {}, edit: { type: 'radio' }, add: { type: 'radio' } },

},

],

## 实现：

Dialog.vue

getItemType(config) {

*return* (config.support[this.inputs.mode] *&&* config.support[this.inputs.mode].type) *||* config.queryType;

},

# 表格的查询类型和编辑弹框名称不一致

## 使用：传入dialogName

columns: [

{

name: '大有Webname',

id: 'Webname',

required: true,

support: {

add: {

dialogName: 'Webname',

},

edit: {

dialogName: 'Webname',

},

},

},

],

## 实现：

Dialog.vue

getLabel(config) {

const label *=* (config.support[this.inputs.mode] *&&* config.support[this.inputs.mode].dialogName) *||* config.name;

*return* label *+* (config.queryType *==* 'title' *?* '' *:* '：');

},

# 添加除了 添加、编辑 其他按钮

## 使用：topBtnsConfig 或 tableBtnsConfig中传入配置添加eventName字段作为自定义事件名，vtable监听即可，同时可以配置按钮样式

docList.vue

topBtnsConfig: [

{

name: '添加文档',

btnType: { isPlain: true, type: 'primary' },

eventName: 'gotoModify',

},

],

<v-table

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

@*gotoModify*="gotoModify"/>

## 实现 ：

handleTopBtnClick(config) {

...

*if* (config.eventName) {

this.$emit(config.eventName, this.tableDataHandled, this.queryParams);

}

},

*// 表格按钮*

handleTableBtnClick(config, rowData) {

...

*if* (config.eventName) {

this.$emit(config.eventName, rowData, this.tableDataHandled);

}

},

# 表格操作列根据数据不一样有不一样的按钮，或者有不同的状态

## 使用：Show传入一个返回布尔值函数或者直接传入布尔值，true时显示。如果传入函数，该函数默认接收当前行的数据。Disabled 同理

tableBtnsConfig: [

{

name: '禁用',

eventName: 'endUse',

show: item => item.state *===* 0,

},

{

name: '启用',

eventName: 'startUse',

show: item => item.state *===* *-*1,

},

],

## 实现

<el-table-column

*v-if*="tableBtnsConfig.length"

*label*="操作"

:*align*="'center'"

*fixed*="right">

<template *slot-scope*="scope">

<div *style*="text-align: center">

<el-button

*v-for*="(oneConfig,index) *in* tableBtnsConfig"

*v-show*="*typeof* oneConfig.show*===*'function'*?*oneConfig.show(scope.row)*:*oneConfig.show*!==*false"

:*key*="index"

*type*="text"

*size*="large"

:*disabled*="*typeof* oneConfig.disabled*===*'function'*?*oneConfig.disabled(scope.row)*:*oneConfig.disabled"

@*click*="()=>handleTableBtnClick(oneConfig,scope.row)">

{{ oneConfig.name }}

</el-button>

</div>

</template>

</el-table-column>

# 表单简单验证

## 使用：直接在columns中的 元素中传入tips即可：

## 实现：

*// 初始化表单规则*

initRule() {

this.inputs.items.forEach(one => {

*if* (one.required) {

this.$set(this.formRules, one.id, [

{ required: true, trigger: 'blur', message: one.tips },

]);

...

}

});

},

# 表单自定义校验

## 使用：传入getValidator以获取到dialog实例中的表单数据

columns: [

{

name: '密码',

id: 'password',

isShow: false,

required: true,

support: ['add', 'edit'],

},

{

name: '确认密码',

id: 'confirmpassword',

isShow: false,

required: true,

support: ['add'],

getValidator(ctx) {

*return* function (rule, value, callback) {

*if* (value *===* '') {

callback(*new* Error('请再次输入密码'));

} *else* *if* (value *!==* ctx.form.password) {

callback(*new* Error('两次输入密码不一致!'));

} *else* {

callback();

}

};

},

},

],

实现：

*// 初始化表单规则*

initRule() {

this.inputs.items.forEach(one => {

*if* (one.required) {

...

*if* (one.getValidator) {

this.formRules[one.id].push({

validator: one.getValidator(this),

trigger: 'blur',

});

}

}

});

},

# 查询条件变化是否自动刷新表格

## 使用：传入searchOnChange（默认true）

## 实现：

*// 查询条件变化，没有点击搜索*

handleQueryChange(val, id) {

...

*if* (this.searchOnChange) {

this.search();

}

},

# 父组件需要根据查询条件做相应处理

## 使用：可以监听queryChange事件，默认传入参数为字段名和当前值

Base.vue

<v-table

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

@*uploadDone*="uploadDone"

@*queryChange*="handlerQueryChange"

@*exportDict*="exportDict">

handlerQueryChange(id, value) {

console.log(value);

this.basicQueryForm.dataValue *=* value;

},

实现：

*// 查询条件变化，没有点击搜索*

handleQueryChange(val, id) {

this.$emit('queryChange', id, val);

...

},

# 查询请求中的某些字段固定为某个值：

## 使用方式：以下2种

1. 传入basicQueryForm

basicQueryForm: {

source: 2,

},

<v-table

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

:*basic-query-form*="basicQueryForm"

:*basic-add-form*="basicQueryForm"/>

1. 监听beforeQuery事件，修改查询的参数

## 实现

search(clearPage *=* false) {

this.loading *=* true;

clearPage *&&* (this.pageNo *=* 1);

const params *=* {

pageSize: this.pageSize,

pagination: this.pageNo,

};

const finalParams *=* Object.assign(params, this.queryParams, this.basicQueryForm);

this.$emit('beforeQuery', finalParams);

...

},

# 后端返回的内容，需要再进行处理再赋值给表格

## 使用

base.vue

<v-table

*ref*="table"

:*table-btns-config*="tableBtnsConfig"

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

@*beforeAssignToTable*="beforeAssignToTable"/>

*// 从后端获取表格数据后进行处理*

beforeAssignToTable(data) {

data.forEach(obj => {

obj.externalInterfaceExtendDTOs.forEach(one => {

obj[one.name] *=* one.value;

obj[`${one.name}Id`] *=* one.id;

});

});

},

## 实现：

search(clearPage *=* false) {

...

this.getData(finalParams)

.then(res => {

this.loading *=* false;

const { model: { records, total } } *=* res;

this.$emit('beforeAssignToTable', records);

this.tableDataHandled *=* records;

this.total *=* total;

})

...

},

# 弹框表单还有自定义的内容，例如其他描述

## 使用：使用slot，传入到vTable中，再传入vTable下的dialog中

topBtnsConfig() {

*return* [

{

name: '字典导出',

btnType: { isPlain: true, type: 'primary' },

eventName: 'exportDict',

},

{

name: '字典导入',

btnType: { isPlain: true, type: 'primary' },

type: 'upload',

action: '/open/supplier/dataRel/import',

eventName: 'uploadDone',

data: this.basicQueryForm,

},

{

type: 'slot',

slotName: 'template',

},

];

},

<v-table

*ref*="brandTable"

:*table-btns-config*="tableBtnsConfig"

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list">

<div *slot*="tips">

<div>

{{ tips }}

</div>

</div>

<div *slot*="template">

<span>

导入为覆盖式更新， 请务必

<a

:*href*="`/${templateName}.xls`"

*class*="downlink">

下载导入模板

</a>

</span>

</div>

</v-table>

## 实现：

vTable.vue

<v-dialog

*ref*="vDialog"

:*inputs*="dataForDialog"

@*dialogClose*="dialogClose"

@*edit*="saveDialogEdit"

@*add*="saveDialogAdd">

<div

*v-for*="(one,index) *in* slotItems"

:*key*="index"

:*slot*="one.slotName">

<slot :*name*="one.slotName" />

</div>

</v-dialog>

computed: {

slotItems() {

*return* this.dataForDialog.items.filter(one => one.queryType *==* 'slot');

},

},

事件从dialog传到vTable中再传到业务组件中

mounted() {

...

this.columns.forEach(column => {

...

*// 初始化查询表单*

*if* (column.support) {

*// 是数组时，是最基础的配置，复杂的配置时是一个对象*

*if* (Array.isArray(column.support)) {

...

} *else* *if* (*typeof* column.support *==* 'object') {

...

const modes *=* ['add', 'edit'];

modes.forEach(mode => {

*if* (column.support[mode] *&&* column.support[mode].eventName) { *// 编辑和新增，事件监听，并且向父组件抛出来*

this.$refs.vDialog.$on(column.support[mode].eventName, obj => {

this.$emit(column.support[mode].eventName, obj);

});

}

});

}

}

});

},

# dialog打开进行数据处理

## 使用：监听vTable的beforeDialogOpen 即可。Dialog关闭处理则监听dialogClose事件

memberManageMent.vue

<v-table

:*table-btns-config*="tableBtnsConfig"

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

:*label-width*="120"

@*endUse*="handleEndUse"

@*beforeDialogOpen*="beforeDialogOpen"

@*startUse*="handleStartUse" />

*// 因为后端返回的密码是md5加密后的，所以前端不应该显示，防止只修改其他字段导致密码丢失*

beforeDialogOpen({ form }, mode) {

*if* (mode *===* 'edit') {

form.password *=* '';

}

},

## 实现

addOrEdit(title, rowData, index) {

...

this.$emit('beforeDialogOpen', this.dataForDialog, mode);

this.dataForDialog.show *=* true;

},

# 编辑或者修改某字段时，需要联动操作

## 使用：columns中添加eventName， dialog事件传到vTable，vTable再传到业务组件，组件监听eventName即可

Changyuan.vue

columns: [

{

name: '天戟信息',

queryType: 'title',

support: ['add', 'edit'],

},

{

name: '供应商ID',

id: 'supplierId',

required: true,

support: {

query: {},

add: {

eventName: 'addIdChange',

},

edit: {

disabled: true,

},

},

},

{

name: '供应商名称',

id: 'supplierName',

required: true,

support: {

query: {},

add: {

disabled: true,

},

edit: {

disabled: true,

},

},

},

],

<v-table

*ref*="table"

:*top-btns-config*="topBtnsConfig"

:*columns*="columns"

:*get-data*="list"

@*addIdChange*="handleIdChange"

@*startUse*="handleStartUse" />

处理函数默认接收一个对象，里面包含dialog中的form表单

*// 供应商id变化处理*

handleIdChange({

form,

}) {

form.supplierName *=* '';

form.corporationName *=* '';

form.corporationId *=* '';

const target *=* this.columns.find(one => one.id *==* 'corporationId');

target.options *=* [];

findSupplierById({ supplierId: form.supplierId }).then(({ model }) => {

*if* (model) {

form.supplierName *=* model.orgName;

} *else* {

this.$message.warning('找不到供应商!');

}

});

...

},

*// 供应商id变化处理*

handleIdChange({

form,

}) {

form.supplierName *=* '';

form.corporationName *=* '';

form.corporationId *=* '';

const target *=* this.columns.find(one => one.id *==* 'corporationId');

target.options *=* [];

findSupplierById({ supplierId: form.supplierId }).then(({ model }) => {

*if* (model) {

form.supplierName *=* model.orgName;

} *else* {

this.$message.warning('找不到供应商!');

}

});

findCorporationList({ supplierId: form.supplierId }).then(({ model }) => {

*if* (model) {

target.options *=* model.map(one => ({

name: one.corporationName,

id: one.id,

}));

}

});

},

## 实现：

mounted() {

this.columns.forEach(column => {

...

*// 初始化查询表单*

*if* (column.support) {

*// 是数组时，是最基础的配置，复杂的配置时是一个对象*

*if* (Array.isArray(column.support)) {

...

} *else* *if* (*typeof* column.support *==* 'object') {

*...*

const modes *=* ['add', 'edit'];

modes.forEach(mode => {

*if* (column.support[mode] *&&* column.support[mode].eventName) { *// 编辑和新增，事件监听，并且向父组件抛出来*

this.$refs.vDialog.$on(column.support[mode].eventName, obj => {

this.$emit(column.support[mode].eventName, obj);

});

}

});

}

}

});

},

<v-dialog

*ref*="vDialog"

:*inputs*="dataForDialog"

@*dialogClose*="dialogClose"

@*edit*="saveDialogEdit"

@*add*="saveDialogAdd">

<div

*v-for*="(one,index) *in* slotItems"

:*key*="index"

:*slot*="one.slotName">

<slot :*name*="one.slotName" />

</div>

</v-dialog>