Vaey2.0

依赖版本

- Vue 2.6.14
- ElementUI 2.15.7
- jquery3.5.1
- eChart 5.2.2
- polyfill.min.js

架构说明

按vue2的语法如果A页面引用了组件B，那么A页面为B页面的父页面，所以frame.js中的 `frame-component` 应该为玄孙，毕竟这个组件是最底下的组件。

frame.js就是画出顶部导航栏，侧边菜单栏、中间tab页，base.js就是中间tab页的底层组件，按我们正常的理解应该说frame.js里面嵌入了base.js，但是生成页面的时候用vue的语法来描述应该base.js是frame.js的父组件。

- 父亲：demo01.html
- 儿子： base-component
    - 儿子的义子1：table-component
    - 儿子的义子2：form-dialog-component
- 孙子：frame-component

代码结构及重点API说明

- component
    - frame
        - frame.css,frame.js中template所用到的样式
        - frame.js,构建后台管理系统的总体页面，如顶部的导航栏，左边的侧边栏，中间的面板
            - 函数说明:
                - createMenu():创建菜单
                - handleCommand(command):处理右上方个人设置
                    - 利用command区分不同的事件
                - openTaskWindow(param):在tab页面中开启一个新的tab页
                    - param:json字符串

        - menuSelect(id, text, url): 菜单选中事件
        - removeTab(targetName):移除tab
        - openTab(name, text, url): 打开一个新的tab页

- base
    - base.css
    - base.js
        - getBtnType(item)
        - getBtnIcon(item)
        - deleteOrNotice(url, fields, message, type, cancel)
        - btnClickHandler(item)
        - conditionFormQuery() 顶部查询栏的查询请求
        - currentRowChange()
        - onTableUpdate()
        - getFormHeaderHeight(): 根据浏览器窗口大小调整table的大小
        - initHeaderCombox() : 初始化头部的url下拉框
        - getSelectedCombox(refValue) : 页面用来选中下拉框的方法
        - handlerUploadButtonBeforeUpload(file, index, item) : 上传文件之前把参数放进去的方法
        - handlerUploadButtonOnSuccess(response, file, fileList, index, item) : 文件上传成功后的回调，通常就是刷新表格
        - setDialogComponentValue
        - getDialogFormData
        - onFileDeleted
        - onItemEvent
        - onDialogValid : formDialog的数据真正提交的地方
        - getQueryParam(): 获取顶部条件查询栏封装好的json对象
        - setFormDialogTableData(ref, key): 设置formdialog table的数据,ref为弹出层的ref,key为当前行对象currentRow中你要选中的数据的key
        - showFormDialog: function(ref, row) 传入ref打开你想要打开的FormDialog

- table
    - table.js
        - refresh() 刷新table数据
        - getData() 获取table数据,其实也是刷新
        - setCurrent() 设置table的数据
        - handleCurrentPageChange 就是把当前选中的行数据映射出去
        - handleSizeChange
        - onRowClick 点击table单行事件

- formDialog
    - formDialog.js
        - show() :显示弹出层

        - onDialogClosed
        - getFormData
        - setComponentValue
        - handleSelectChange
        - checkTimeRange
        - processCascaderData
        - cancel：点取消
        - confirm：点确定
        - hide：在base.js的onDialogValid执行完post请求后会调用这个函数
        - checkAllNodes
        - uncheckAllNodes
        - getTreeNode
        - setFormDialogTableData(key): 设置formdialog table的数据,key为当前行对象currentRow中你要选中的数据的key
- common
- html
    - index.html 引用frame.js，根据传入的userName定义顶部右边的用户名、tab页面打开新的tab页的传递函数这这里: `window.addEventListener('message', function (e) {...});`
    - default.html 默认的首页，通常不去修改它
- img–放各种图片
- lib–存放各种类库，如vue、jquery、elementui
- css
    - common.css–顶部导航栏有间隙,引用它可以取消间隙

完整代码骨架

```
<!--先导入头部js文件-->
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>demo01</title>
    <link rel="stylesheet" href="../../lib/element-ui/element.css">
    <script src="../../common/constant.js"></script>
    <script src="../../common/utils.js"></script>
    <script src="../../lib/vue/vue.js"></script>
    <script src="../../lib/element-ui/element.js"></script>
    <script src="../../lib/jquery/jquery3.5.1.js"></script>
    <script src="../../component/base/base.js"></script>
    <link rel="stylesheet" href="../../component/table/table.css">
    <script src="../../component/table/table.js"></script>
    <script src="../../component/formDialog/formDialog.js"></script>
    <link rel="stylesheet" href="../../component/formDialog/formDialog.css">
</head>
<body>
  <div id="app">
```

```
    <base-component ref="base"
                    @current-row-change="currentRowChange"
                    @query-param="queryParam"
                    :dialogs="dialogs"
                    :table="table"
                    :form="form">
    </base-component>
</div>
<script>
  var GLOBAL_ENUM = {
    SEX: {
      CONVERT_TEXT: 'sexText',
      CONVERT: {
        1: '男',
        2: '女',
      }
    }
  }
  var vueObj = new Vue({
                el: '#app',
                            data: function() {
                  return {
                    currentRow: null,
                form: {
                  //详见tab页顶部各类查询条件栏
                      buttons: [],
                  uploadButtons: [],
                  inputs: [],
                  radios: [],
                      dates: [],
                  selects: []
                },
                table: {
                  //详见tab页中的table
                },
                dialogs:[
                      //详见form弹出层
                ],
              }
            },
            mounted: function (){
              //testSelected其实就是上面指定的id，切记不能重复
              //这里要回调base.js的函数才能绑定值,这是固定套路写法
              this.$refs.base.getSelectedCombox("testSelected")
                          },
                              methods: {
              currentRowChange: function (val) {
                    this.currentRow = val
                  },
              queryParam: function(val) {
                  //console.log(val)
              },
              handleSelectChange:function (json) {

                console.log(json.field);
                console.log(json.value);

                var param = {
```

```
                'ref':'addDialog',
                'fields': {
                    "title": "下拉框赋值改变标题",
                    "userId": "下拉框改变userId"
                }
            }
            //方法一
            this.$refs.base.setDialogComponentValue(param)
            //方法二
            var formData = this.$refs.base.getDialogFormData('addDialog')
            formData['userPhone'] = 13799118590
                    }
        }
    })
  </script>
</body
```

总体框架说明及示例

总体框架是指顶部导航栏、右边侧边栏、中间面板、配置右上方个人中心。

画出总体框架

代码如下:

```
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>互医健康UI系统展示</title>
    <link rel="stylesheet" href="../lib/element-ui/element.css">
    <link rel="stylesheet" href="../css/common.css">
    <script src="../lib/babel-polyfill/polyfill.min.js"></script>
    <script src="../lib/vue/vue.js"></script>
    <script src="../lib/element-ui/element.js"></script>
    <script src="../lib/jquery/jquery3.5.1.js"></script>
    <link rel="stylesheet" href="../component/frame/frame.css">
    <script src="../component/frame/frame.js"></script>
</head>
<body style="height: 100%;">
    <div id="app">
        <frame-component
                ref="framecomponent"
                :user-name="userName"
                logo-txt="互医健康"
                menu-url="/BNWeb/menuAction/createMenu"
                default-src="default.html"
                img-url="../img/hyjk.png"
        ></frame-component>
    </div>
</body>
<script>
    var vm = new Vue({
        el: '#app',
        data: function () {
```

```
            var _this = this
            return {
                userName: '可以修改用户名,我在index.html中'
            }
        },
        create: function () {
            var name = decodeURI(getUrlParams('userName'));
            if(name !== "undefined") {
                this.userName = name
            }
        },
        methods: {

        }
    });
    function getUrlParams(name) {
        var param = window.location.search === "" ? window.location.hash : window.location.search;
        if (param.length > 1) param = param.substring(param.indexOf("?") + 1).split("&");
        var paramObj = {};
        for (var i = 0; i < param.length; i++) {
            var p = param[i].split("=");
            paramObj[p[0]] = p[1];
        }
        if (name) return paramObj[name];
        return paramObj;
    }
    window.addEventListener('message', function (e) {

        var param = e.data;

        switch (param.type) {
            //1就是打开一个windows
            case 1:
                vm.$refs.framecomponent.openTaskWindow(param)
                //vm.$refs.framecomponent.openTaskWindow(param.opId, param.title, param.nodeName, param.nodeKey)
                break;
            default:
                break;
        }
    });
</script>
</html>
```

index.html代码如上面给出所示，通常不修改，毕竟侧边栏，导航栏正常是固定的，在使用frame-componet的时候会要求传递一些参数：

- logo-txt 通常指项目名称，最多四个字
- menu-url 侧边栏菜单的action
- default-src 首页路径，建议写default.html
- img-url 单位的logo
- user-name 当前用户名，这个用户名要绑定变量，用的是data函数的userName，可以根据浏览器的参数解析例如http://localhost:8081/vaey?userName=yzl,根据这个链接会解析出yzl,当然不传也是可以的，或者我们在可以data函数中使用ajax跟服务器请求用户名然后再赋值给userName也是可以的

侧边栏菜单

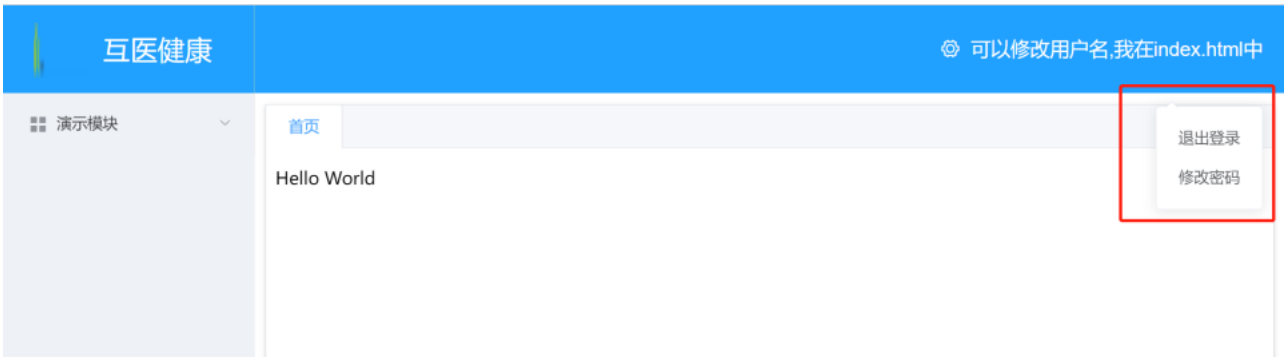
侧边栏菜单的生成函数在frame.js中的 `createMenu` 函数中,后台传递给前端的数据格式，也就是根据menu-url生成的数据格式如下:

```
var d = [{
  ”text”: ”演示模块”,
  ”url”: null,
  ”opId”: ”101”,
  ”parentId”: ”21”,
  ”leaf”: false,
  ”type”: 1,
  ”keyCode”: null,
  ”icon”: null,
  ”children”: [{
    ”text”: ”表格页面”,
    ”url”: ”demo/demo1.html”,
    ”opId”: ”1001”,
    ”parentId”: ”101”,
    ”leaf”: true,
    ”type”: 2,
    ”keyCode”: null,
    ”icon”: null,
    ”children”: []
  },{
    ”text”: ”表格页面2”,
    ”url”: ”demo/demo2.html”,
    ”opId”: ”1004”,
    ”parentId”: ”101”,
    ”leaf”: true,
    ”type”: 2,
    ”keyCode”: null,
    ”icon”: null,
    ”children”: []
  },{}//...可以有更多]
            },{}//..可以有更多];
```

- text 菜单名称
- url 如果是2级菜单通过url可以访问页面，仅能有二级菜单
- leaf true表示二级菜单,false表示一级菜单栏目
- children 放子菜单

个人设置(修改密码和登出)

个人设置如下图所示，通常有退出登录，修改密码，如果有需要可以再加



代码在frame.js中,html代码如下，显然如果需要添加则在修改密码下发继续增加 `<el-dropdown-item command="自定义key">自定义文字</el-dropdown-item>` ，具体的处理函数为 `@command="handleCommand"` ,也就是 `handleCommand`

```
<el-dropdown @command="handleCommand" trigger="hover" style="color: #fff; cursor: pointer; font-size: 18px;">
  <span class="el-dropdown-link">
    <i class="el-icon-setting" style="margin-right: 5px"></i>
    <span>{{userName}}</span>
  </span>
  <el-dropdown-menu slot="dropdown">
    <el-dropdown-item command="logout">退出登录</el-dropdown-item>
    <el-dropdown-item command="editPwd">修改密码</el-dropdown-item>
  </el-dropdown-menu>
</el-dropdown>
```

假如command设为 `selfKey` ，则我们需要在handleCommand中添加相关的方法

```
handleCommand: function (command) {
            //处理右上方个人设置
            if(command == "logout") {
                                                    //...
            }else if(command == "editPwd") {
                //...
            }else if(command == "selfKey"){
                    //...添加自定义的方法即可
            }else{
                //..
            }
        }
```

alert提示框和事件确认框

```
//alert弹出提示框,this是vue的对象
this.$message.warning("请选择一条记录");

//事件确认框,this是vue对象
this.$confirm("确认框,可以选择确定和取消", "提示", {
    confirmButtonText: "确定",
    cancelButtonText: "取消",
    type: "warning"
    }).then(() => {
                //点击确定的回调事件
        this.$message({
            type: "success",
            message: "点击确定!"
        });
```

```
    }).catch(() => {
            //点击取消的回调事件
        this.$message({
            type: ”info”,
            message: ”已取消退出”
        });
    });
```

打开一个新的tab页面

在index.html有如下代码,这段是调用子组件framecomponent的openTaskWindow方法,它会把json参数直接带到frame.js对应的方法,这里注意type要为1,1是打开一个新窗口

```
window.addEventListener(’message’, function (e) {

    var param = e.data;

    switch (param.type) {
        //1就是打开一个windows
        case 1:
            vm.$refs.framecomponent.openTaskWindow(param)
            break;
        default:
            break;
    }
})
```

比如我们期望在页面A启用一个方法然后能打开一个新的页面B,那么我们需要在页面A写的点击方法写如下代码中的 `openTaskWindow` ,这样就可以调用到上面的父组件的方法,前面四个参数title、url、callUrl、type是必填:
title:tab页的标签
url:根据这个url去获取页面，url和callurl只能填写一个，如果都写则以callurl为准
callUrl:根据这个callUrl再结合其余的参数从后台拿回两个参数一个是title,一个是url,title就是tab页名字,url就是跟上一个参数url一个意思,格式:

其他参数: 其他参数是自定义，通常是结合callUrl在后台判断所用

```
[{”code”:100,”datas”:[{”title”:”tab名字”, ”url”:””}]}]
```

页面A中的 `openTaskWindow` 的函数案例一:

```
openTaskWindow: function(event) {
    click: function() {
        var param = {
            title: ”测试标题”,
            url: ”/BNWeb/configMenuAction/unsatisfactory.do”,
            callUrl:””,
            type: 1,
            nodeKey:”temp”,
        };
        let targetOrigin = ”*”;
        window.parent.postMessage(param, targetOrigin);
                }
}
```

案例二：callUrl配合其他参数在后台使用：

```
openTaskWindow: function(event) {
  //子组件调用父组件并弹出一个新的tab页
  var param = {
    title: event.target.dataset.title,
    nodeName: event.target.dataset.nodename,
    opId: event.target.dataset.opid,
    nodeKey: event.target.dataset.nodekey,
    callUrl:"/BNWeb/frameAction/getFlowTitleAndUrl.do",
    type: 1
  };
  let targetOrigin="*";
  window.parent.postMessage(param, targetOrigin);
}
```

tab页顶部各类查询条件栏

button–直接action通知后台(如delete、notice)

delete和notice是类似的后台调用的函数是相同，就是图片不同 `click` 和 `cancel` 是可以不写的，完整代码查看demo01.html，代码片段如下，multi功能可以配合目录tab页面中的table的多选提交一起参考

```
var vueObj = new Vue({
  el: '#app',
  data: function() {
          return {
            form: {
              buttons: [
          {type: 'delete', url: "/deleteulr.do", fields: ['opId'], message:"弹出删除的提示,有默认提示可以不填写哈", label: '按钮四'},
          {
            type: 'notice',
            url: "/notice_ulr.do",
            fields: ['opId'],
            message:"弹出提示的提示,默认是删除,最好还是要写提示",
            label: '通知后台',
            cancel(){
                    //可以不写这个函数
                    console.log("通知的取消提示")
            },

          },{
            type: 'delete',
            url: "/deleteulr.do",
            fields: ['opId'],
            multi: true,//启用多选功能，此时点击该按钮不再以当前选择row为数据，而是以选中的勾选框为数据
            label: '按钮6',
            cancel(){
                    //可以不写这个函数
                    console.log("删除的取消提示")
            },
          },{
                  type: 'warning',
            label: 'test',
            cancel() {},
            click() {}
```

```
        }
      ]
    }
  }
}

});
```

upload-顶部查询栏上传按钮

完整代码查看demo01.html，上传成功后会刷新table

```
var vueObj = new Vue({
  el: '#app',
  data: function() {
          return {
            form: {
        uploadButtons: [
          {
            type: 'add',
            label: '点击上传文件1',
            name: 'test',//文件上传时对应的名称
            action: 'http://127.0.0.1:12002/iom/api/uploadTest',//后台action
            accept: '.xls,.xlsx',//允许上传的类型
            size: 1,//文件大小(默认是1,目前仅为1)
            required: true,
            fields: ['opId', 'title'],//如果要传参数则当前行为必填项,即required为true
            onSuccess: function() {
              //文件上传成功之后的回调函数,如果没有上传成功则不会回调
              //如果上传成功会强制刷新一次table
              //通常onSuccess是可以不写的
              alert("ok")
            }
          },
          {
            type: 'add',
            name: 'test',//文件上传时对应的名称
            label: '点击上传文件2',
            action: 'http://127.0.0.1:12002/iom/api/uploadTest',//后台action
            accept: '.pdf',//允许上传的类型
            size: 1,//文件大小(默认是1,目前仅为1)
            required: true,
            fields: ['opId', 'title']
          }
        ],
      }
    }
  }
});
```

uploadButtons回调说明,这个按钮除了 `onSuccess` 这个无参函数外，还有一个 `onCall` 函数可以提供调用，当我们因为某些特殊原因必须获取后台的返回参数进行操作的时候就可以使用这个方法，先看使用说明(完整代码参考demo01.html):

- 首先在base-componet中添加一个 `@test-call="testCall"`
- 在uploadButtons中添加 `onCall: 'test-call'` ，这里的 `test-call` 就是对应base-component中的 `@test-call`

- 最后在methods中添加 `testCall: function(response, file, fileList, index)` ，其实就是把这个testCall函数映射给base组件让它可以调用到

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<!--先导入头部js文件,省略不写-->
</head>
<body>
  <div id="app">
     <base-component ref="base"
                    @current-row-change="currentRowChange"
                    @query-param="queryParam"
                    @test-call="testCall"
                    :dialogs="dialogs"
                    :table="table"
                    :form="form">
     </base-component>
  </div>
  <script>

    var vueObj = new Vue({
                  el: '#app',
                              data: function() {
                    return {
                      currentRow: null,
                  form: {
                    //详见tab页顶部各类查询条件栏
                         buttons: [],
                    uploadButtons: [                  {
                           type: 'add',
                           label: '点击上传文件1',
                           name: 'test',//文件上传时对应的名称
                           action: 'http://localhost:12006/WS/api/uploadTest',//后台action
                           accept: '.xls,.xlsx',//允许上传的类型
                           size: 1,//文件大小(默认是1,目前仅为1)
                           required: true,
                           fields: ['opId', 'title'],//如果要传参数则当前行为必填项
                           onSuccess: function() {
                               alert("ok")
                           },
                           onCall: 'test-call'
                       },],
                    inputs: [],
                    radios: [],
                         dates: [],
                    selects: []
                  },
                  table: {
                    //详见tab页中的table
                  },
                  dialogs:[
                         //详见form弹出层
                  ],
                 }
               },
               mounted: function (){
```

```
            //testSelected其实就是上面指定的id，切记不能重复
            //这里要回调base.js的函数才能绑定值,这是固定套路写法
            this.$refs.base.getSelectedCombox("testSelected")
                    },
                        methods: {
            currentRowChange: function (val) {
                this.currentRow = val
                },
            queryParam: function(val) {
                //console.log(val)
            },
            testCall: function(response, file, fileList, index) {
                //uploadButtons的回调函数
            }
          }
    })
  </script>
</body
```

如果有使用 `onCall` 函数框架就不再主动帮你刷新列表了，如果有需要则自行刷新列表

inputs组件

输入框组件

```
var vueObj = new Vue({
  el: '#app',
  data: function() {
          return {
            form: {
        inputs: [
          {name: 'inputFieldName', placeholder: '我是输入框', value: '输入test'},
        ],
      }
    }
  }

});
```

如上代码所示可以直接写value,这个会作为默认值传递过去，在base.js的data函数的 `initQueryParams`

radio组件

注意下，这个radio是自己传递 `onChange` 的，可以根据情况变化form中的数据。完整代码查看demo01.html

```
new Vue({
    el: '#vue',
    created: function() {},
    data: function() {
        var _this = this;
        return {
            form: {
                buttons: [],
                inputs: [],
```

```
            radios: [{
                    name: 'area',
                    value: 'city',
                    onChange: function (val, form) {
                        if (val === 'city') {
                            form.cityCode = '';
                            form.type = 1;
                        } else {
                            form.cityCode = GLOBAL_OPTS.CITY[0].value;
                            form.type = 2;
                        }
                    },
                    options: [{
                        label: '地级市',
                        value: 'city'
                    }, {
                        label: '区县',
                        value: 'county'
                    }]
                },
                //按机构统计
                {
                    name: 'type',
                    value: 1,
                    onChange: function (val, form) {
                        form.type = val;
                    },
                    dynamic: function (currentRow, form) {
                        return form.area === 'county'
                    },
                    options: [{
                        label: '按地区',
                        value: 2
                    }, {
                        label: '按机构',
                        value: 3
                    }]
                }
                ],
            selects: [],
            dates: []
        }
    }
}

})
```

可以直接写value,类似于input,这个会作为默认值传递过去，在base.js的data函数的 `initQueryParams`

select下拉框查询

```
{
    name: '', //提交时的字段名
    value: 0,
    clearable: false, //true:可清除,false:不可清除
    //id: 'testSelected',
```

```
    //isRemoteSelected: true,
    placeholder: '',
    options: [{}, ...], //选项 {label: '显示内容', value: 值} //如果要获取远程数据,可以在create中选获取到然后把值赋给这个options即可
    optionsUrl:'url',//根据url获取值,和options不可以同时存在,[{"name":"", "value":""},{...}]
    onChange: function(value, queryParams, currentRow){ //change事件监听函数
        //value 当前选中值
        //queryParams 整个form查询表单的当前值
        //currentRow 当前选中的table行
    }
}
```

举个例子:

```
{
            name: 'adviceStatus',
            value: 0,
                    placeholder: '收费状态',
      clearable: false,
      options: [{
        value: 0,
        label: '待收费'
      }, {
        value: 1,
        label: '已收费'
      },{
        value: 2,
        label: '作废'
      },{
        value: 3,
        label: '回退'
      }]
}
```

下拉查询联动用法:示例：此处countyCode区县的opitons要根据cityCode地级市的值而定。在cityCode中配置onChange函数监听change事件，然后根据当前选中值，手动将所需的options绑定到countyCode上。可以直接写value,类似于input,这个会作为默认值传递过去，在base.js的data函数的 `initQueryParams`

```
data: function () {
    var _this = this

    return {
        form: {
            selects: [
                    {
                name: 'cityCode',
                placeholder: '请选择所属地级市',
                value: '',
                clearable: true,
                onChange: function(val, queryParams, currentRow) {
                        _this.form.selects[1].options = GLOBAL_OPTS.CITY_COUNTY[val];
                        queryParams.countyCode = '';
                            },
                options: GLOBAL_OPTS.CITY},
                    {
                name: 'countyCode',
                placeholder: '请选择所属区县',
                value: '',
```

```
            clearable: true,
            options: []}
        ],
        ...
    }
}
```

如果需要将label的值提交，那么属性name需要在结尾加上Select关键字，例如 hospitalSelect，此时会提交两个字段，分别是hospitalSelectValue对应选项的value值，以及hospitalSelectName对应选项的label值。

```
selects: [
        {
            name: 'hospitalSelect',
            placeholder: '我是下拉框',
            options: [{
                value: 1,
                label: '医院1'
                }, {
                value: 2,
                label: '医院2'
                }]
        },
],
```

远程选中下拉框的值，意思就是组件初始化后直接跟后台代码请求数据，然后后台指定一个值前台选中，后台格式如下:

```
{
  "value":"v2",//v2就是说选中value值为v2的对象
  "datas": [{
    "name":"查询框远程绑定值下拉框1",
    "value": "v1"
  },{
    "name":"查询框远程绑定值下拉框2",
    "value": "v2"
  }]
}
```

远程文字搜索下拉框

这个通常用在数据量很大的场景

```
var vueObj = new Vue({
  el: '#vue',
  data: function () {
    ...//省略代码
    return {
      ...//省略代码
      form:{
        selects:[{
          name: 'testInputSelect',
          placeholder: '输入下拉框',
          clearable: true,
          options: [],//一定要有这个
          props: {
            filterable: true,
            remote: true,
            //multiple: true, //不支持多个查询
```

```
            'reserve-keyword': true,
            'remote-method': function(query) {
              //一定要针对当前的这个options对象进行操作
              _this.form.selects[0].options = [{label: 'tt', value: '111'},{label: 'tt2', value: '222'}]
            }
          },
        }
                ]
      }
    }
  },
    })
```

远程选中下拉框的值

关键源码查阅base.js的 `initHeaderCombox、getSelectedCombox`

```
  var vueObj = new Vue({
    el: '#vue',
    data: function () {
              ...//省略代码
      return {
          ...//省略代码
               form:{
                 selects:[{
               //如果需要后台指定下拉框的值,这里的id和isRemoteSelected一定要写上
               //id不能重复谢谢
               name: 'selected',
               placeholder: '我是获取远程数据并选中的下拉框',
               optionsUrl: "./org3.json",
               id: 'testSelected',
               isRemoteSelected: true,
              }
            ]
          }
      }
    },
    mounted:function () {
        //testSelected其实就是上面指定的id，切记不能重复
       //这里要回调base.js的函数才能绑定值,这是固定套路写法
        this.$refs.base.getSelectedCombox("testSelected")
    },
  })
```

顶部条件查询下拉框选中值解析：清楚了解临时存储值的对象 `selectcommbox`

- 关键代码在base.js中,首先在 `create` 函数中如下代码:

```
if(item.isRemoteSelected) {
       //把该id的值赋给全局的json对象,这里id是不能重复的
  _this.selectCommbox[item.id] = resData.value;
  //这里的queryParams是传递给table.js的参数,所以要加在这里要不然后台没有选中值的参数
  _this.queryParams[item.name] = resData.value;
  resData = resData.datas;
}
```

然后在页面中调用的代码 `this.$refs.base.getSelectedCombox("id")` 解析:

```
//这段代码也在base.js中，意思就是选中value值
getSelectedCombox:function (refValue) {
  var component = this.$refs[refValue][0];
  component.value = this.selectCommbox[refValue];
  //移除相关的Key
  delete this.selectCommbox[refValue]
},
```

dates时间组件

```
dates: [{
    name: '-daterange',
    type: 'daterange',
    startPlaceholder: '事件开始日期',
    endPlaceholder: '事件结束日期'
}]
```

可以直接写value,类似于input,这个会作为默认值传递过去，在base.js的data函数的 `initQueryParams`

后台会以startTime和endTime作为key把时间传到后台去
再来一个具体的example，通常就是把dates加入到form里面去,然后就出现两个时间框组件

```
new Vue({
    el: '#vue',
    created: function() {},
    data: function() {
        var _this = this;
        return {
            form: {
                buttons: [],
                inputs: [],
                radios: [],
                selects: [],
                dates: [{
                            name: '-daterange',
                            type: 'daterange',
                            startPlaceholder: '事件开始日期',
                            endPlaceholder: '事件结束日期'
                        }, ]
            }
        }
    }

})
```

如果需要通过自定义按钮的点击时间获取时间组件的值,可以这样做:
首先需要在base-component标注@time-change,然后函数名可以自定义,这里定义为timeChange

```
<div id="vue">
    <base-component ref="base" :form="form" :table="table" :dialogs="dialogs" @current-row-change="
currentRowChange"
                    @time-change="timeChange"
                    @table-update="onTableUpdate">
        <template slot="tableRightContainer">
```

```
        <div id="chart" style="width: 100%; height: 100%;box-sizing: border-box;padding-top: 25px;"></div>
      </template>
    </base-component>
</div>
```

时间组件主要解析地方是在base.js的watch里面

动态获取顶部条件栏name对应的参数值

首先在base组件先配置一个代码段: `@query-param="queryParam"` ，queryParam是函数名,可以根据自己的习惯编写

```
<div id="app">
    <base-component ref="base"
                    @current-row-change="currentRowChange"
                    @query-param="queryParam"
                    :dialogs="dialogs"
                    :table="table"
                    :form="form">
    </base-component>
</div>
```

然后在methods中添加queryParam函数,这样一来当查询栏的参数发生变化时这里就可以监听到了

```
methods: {
  queryParam: function(val) {
    //base.js中有emit会调用这个参数,并传递val过来
    console.log(val)
                console.log(val['key']) //key就是我们自己定义的name值
        },
}
```

上面的代码时被动通知，那我们如何实时的调用呢?

```
methods: {
  getBaseQueryParam: function () {
                //这个getQueryParam时在base.js中定义好的参数
                  var query = this.$refs.base.getQueryParam()
                  console.log(query)
                console.log(query['key'])
  },
}
```

查询栏与后台交互源码分析

其实就是在base.js中的data里面有一个initQueryParams的对象，然后去遍历我们写的form对象，排除button，其余类型的模型都是 `queryParams[item.name]` ,其实就是根据name把值放进去，到这里就封装好参数了

```
    data: function() {
        var _this = this
        var initQueryParams = {}
        //初始化默认值
        Object.keys(_this.form).forEach(function (key) {
            if(key !== 'buttons' && key !=='uploadButton') {
                _this.form[key].forEach(function (item) {
                    if(item.value != null) { //有默认值
```

```
                    initQueryParams[item.name] = item.value
                } else {
                    initQueryParams[item.name] = '' //空值，响应化
                }
            })
        }
    })
    return {
        queryParams: initQueryParams,
        //...
    }
},
```

tab页中的table

多选提交

buttons的button添加一个属性 `multi:true` ，框架会自动提交数据

```
buttons:[{
  type: 'delete',
  multi:true,
  url: "http://localhost:12006/WS/api/deleteurlArray",
  fields: ['opId', 'title'],
  label: '多选删除'
}]
```

然后后台代码

```
@GetMapping(value = "api/deleteurlArray", produces = "application/json; charset=utf-8")
public String deleteurlArray(String[] opId, String[] title) {
    //return renderQuerySuccessList(1);
    return renderFailureList("我草");
}
```

如果需要获取多选的数据自己再进行操作的,要向base组件添加代码: `@multi-row-change="multiRowChange"`

```
<base-component ref="base"
                @current-row-change="currentRowChange"
                @multi-row-change="multiRowChange"
                @query-param="queryParam"
                @test-call="testCall"
                :dialogs="dialogs"
                :table="table"
                :form="form">
</base-component>
```

然后在data对象中添加一个multiRow，当然可以自定义

```
var vueObj = new Vue({
    el: '#app',
    data: function() {
      return {
        multiRow: null,
```

```
        }
      }
  })
```

最后在methods中添加方法就可以获取到数组数据

```
methods: {
                multiRowChange: function (array) {
            this.multiRow = array
          },
}
```

返回格式

```
{
      ”total”: 2,
      ”code”: 100,
      ”success”: true,
      ”datas”: [{
            ”opId”: ”test1”,
            ”title”: ”标题”,
            ”createTime”: ”2021-02-21”,
            ”sex”: 1,
            ”address”: ”福建省福州市”,
            ”author”: ”互医健康”,
            ”phone”: ”13799118590”,
            ”numberSelect”: ”1”
      }, {
            ”opId”: ”test2”,
            ”title”: ”标题”,
            ”createTime”: ”2021-02-21”,
            ”sex”: 1,
            ”address”: ”福建省福州市”,
            ”author”: ”互医健康”,
            ”phone”: ”13799118590”,
            ”numberSelect”: ”2”
      }],
      ”desc”: ”成功”
}
```

示例代码

完整代码参考demo01.html，这里最大的改变就是枚举。

```
new Vue({
  el: '#vue',
  created: function() {},
  data: function() {
    //这里枚举进行了修改,CONVERT_TEXT是指转换后枚举对应的key,
    //原来sex的值是1和2，现在是sexText的值为男、女
    //可以根据实际需要更改CONVERT_TEXT的值
    var GLOBAL_ENUM = {
      SEX: {
```

```
      CONVERT_TEXT: 'sexText',
      CONVERT: {
        1: '男',
        2: '女',
      }
    }
  }
  var _this = this;
  return {
    form: {

    },
    table: {
      url: "tabledemodata.json",
      //url: "http://localhost:12006/WS/api/queryTest",
      //正常情况下manual的默认值为false,然后框架的getData()函数会根据配置的url取table的数据
      //当有需要自定义获取数据的时候就按照下面的配置,请注意因为暂时没有适用场景
      //不建议开启下面的方法,有需要建议联系开发人员补充功能
      // manual: true,
      // handler: function() {
      //      console.log("handler")
      // },
      //如果真的要处理datas，要记得把datas在返回回去要不然就没数据咯
      //dataProcess: function (datas) {
      //    return datas
      //},
      cols: [
        {
          prop: "title",
          label: "标题",
          width: 100
        },
        {
          prop: "createTime",
          label: "创建时间",
        },
        {
          prop: "sexText",
          label: "性别"
        },{
          prop: "address",
          label: "地址"
        },{
          prop: "author",
          label: "作者"
        }
      ],
      enumMap: {
        sex: GLOBAL_ENUM.SEX
      },
      params: {
        'userName': 'yzl'
      },
      sort:'title' //根据某个字段排序
    }
  }
}
```

})

后台返回的格式

{”total”:10, ”code”:100, ”desc”,”描述”, message:”信息”,”datas”:[{...},{...}...]}

table的查询流程

- 在base.js中会封装好参数 `queryParams` ,这个参数会被传递到table.js中
- table.js中的 `created` 会执行 `getData` ,这里就是第一次执行数据的地方
- 在base.js的 `conditionFormQuery` 会调用 `getData` ,然后再table.js中会监听 `pageSize、currentPage` ， 这两个参数一动就会重新刷新数据，就是调用 `getData`

table行点击事件

简而言之在组件传递一个 `@row-click=”rowClick”` ,然后在method中写一个rowClick即可

```
<base-component ref=”base”
                @current-row-change=”currentRowChange”
                @multi-row-change=”multiRowChange”
                @row-click=”rowClick”
                @query-param=”queryParam”
                @test-call=”testCall”
                :dialogs=”dialogs”
                :table=”table”
                :form=”form”>
</base-component>
```

通过点击table的单行数据直接打开formDialog

这个必须要配合行点击事件来做，上面有提到的rowClick方法中调用base组件的 `showFormDialog` ， 传递的参数就是你自己写的弹出层的ref

```
methods: {
    rowClick: function (row) {
        this.$refs.base.showFormDialog(”addDialog”, row)
    },
}
```

源码解析

动态调整table大小

首先在 `el-table` 中挂在一个 `TheStyle` 的样式对象

```
<el-table :style=”TheStyle”
                                  //...
></el-table>
```

这个 `min-height` 就是我们要调整的大小

```
data: function() {
        return {
    TheStyle: {
      'min-height':'60px',
      'width': '100%',
      'height': '100%',
      'overflow': 'auto'
    }
  }
}
```

在table.js的mounted中有如下代码:

```
    mounted: function () {
        this.$emit('get-form-header-height')
        var vm = this
        $(window).resize(function (){
            vm.$emit('get-form-header-height')
        });
    },
```

不难看出这里是调用base.js中注册的 `@get-form-header-height` ，接下来再看base.js的代码,应该很清晰了，就不解释了

```
getFormHeaderHeight: function(obj) {
  //tab的高度-头部高度-底部分页高度(就是最后的45) 就是 中间table的高度
  let height = window.innerHeight - this.$refs['formHeader'].clientHeight - 45;
  height = height > 60 ? height : 60
  this.$refs.globalSingleTable.TheStyle["min-height"] = height + "px"
},
```

table多表头写法

如果只有1行表头，则按原来的写法，不需要添加children；要再多加一个children，下方的案例是3行表头，所以有2个children，目前table.js支持4行表头，如果有需要再增加表头，则将table.js的el-table-column再多写一层：

```
var TABLE_TPL = '  <div>\n' +
    '     <div class="tbl-wrapper">\n' +
    '         <el-table ref="_table" v-bind="props" :data="rows" :row-class-name="rowClassName" stripe border highlight-
current-row @current-change="_handleCurrentRowChange" @selection-change="_handleSelectionChange" @row-click="
onRowClick" style="width: 100%;height: 100%;overflow: auto;">\n' +
    ' <el-table-column\n' +
    '       type="selection"\n' +
    '       width="55">\n' +
    '     </el-table-column>' +
    ' <slot name="tableColumn"></slot>\n' +
    '             <el-table-column v-for="col in cols" :key="col.prop" :prop="col.prop" :label="typeof col.label === \'object\' ?
col.label.value : col.label" :width="col.width" >\n' +
    '                               <el-table-column v-if="col.children||col.children.length>0" v-for="item1 in col.children" \n' +
    '                               :label="item1.label" :prop="item1.prop" :width="item1.width" > \n' +
    '                                       <el-table-column v-if="item1.children||item1.children.length>0" v-for="item2 in item1.
children" \n' +
```

```
    ’                                   :label=”item2.label” :prop=”item2.prop” :width=”item2.width” > \n’ +
        ’                                   <el-table-column v-if=”item2.children||item2.children.length>0” v-for=”item3 in item2.
children” \n’ +
        ’                                   :label=”item3.label” :prop=”item3.prop” :width=”item3.width” > \n’ +
        ’                                   </el-table-column>\n’ +
        ’                                   </el-table-column>\n’ +
        ’                               </el-table-column>\n’ +
        ’           </el-table-column>\n’ +
    ’       </el-table>\n’ +
    ’    </div>\n’ +
    ’    <el-pagination\n’ +
    ’           @size-change=”_handleSizeChange”\n’ +
    ’           @current-change=”_handleCurrentPageChange”\n’ +
    ’           :current-page=”currentPage”\n’ +
    ’           :page-sizes=”pageSizes || [10, 25, 50, 100]”\n’ +
    ’           :page-size=”pageSize”\n’ +
    ’           layout=”total, sizes, prev, pager, next, jumper”\n’ +
    ’           :total=”total”>\n’ +
    ’           :manual=”manual”>\n’ +
    ’           :handler=”handler”>\n’ +
    ’    </el-pagination>\n’ +
    ’</div>’

var v = new Vue({

    data: function() {
        return {
            table: {
                url: ”xxx/xxx.do”,//请求后台数据的地址
                cols: [{
                                                    prop: ””,
                                                    label: ”500mg/L含氯消毒液”,
                                                    children: [{
                                                            prop: ””,
                                                            label: ”台面”,
                                                            children: [{
                                                                    prop: ”recordTime”,
                                                                    label: ”日期”,
                                                            },
                                                            {
                                                                    prop: ”tableTopMorning”,
                                                                    label: ”上午”,
                                                            },
                                                            {
                                                                    prop: ”tableTopAfternoon”,
                                                                    label: ”下午”,
                                                            },
                                                            {
                                                                    prop: ”tableTopOperatorUserName”,
                                                                    label: ”消毒人员”,
                                                            }
                                                            ]
                                                    }, {
                                                            prop: ””,
                                                            label: ”地面”,
                                                            children: [{
                                                                    prop: ”recordTime”,
                                                                    label: ”日期”,
```

```
                                                },
                                                {
                                                        prop: ”groundMorning”,
                                                        label: ”上午”,
                                                },
                                                {
                                                        prop: ”groundAfternoon”,
                                                        label: ”下午”,
                                                },
                                                {
                                                        prop: ”groundOperatorUserName”,
                                                        label: ”消毒人员”,
                                                }
                                        ]
                                    }]
                                }],
            enumMap: {
                sex: GLOBAL_ENUM.EVENT_TYPE//性别后台传1和2,这里映射到枚举中变为男和女
            },
            dataProcess: function (datas) {//这里可以回调到数据,方便在显示前先处理一下数据
                console.log('dataProcess, ', datas)
                return datas
            },
            //正常情况下manual的默认值为false,然后框架的getData()函数会根据配置的url取table的数据
            //当有需要自定义获取数据的时候就按照下面的配置,请注意因为暂时没有适用场景
            //不建议开启下面的方法,有需要建议联系开发人员补充功能
            // manual:true,
            // handler(_this) {
            //    _this.queryObj //queryObj就是封装好的条件参数

            //    ...一系列请求后台数据的代码
            //  //最终要把把获得的数据放到rows、total中
                //  _this.rows = resData.datas
                // _this.total = resData.total
            // },
            // sort: '[{”property”:”create_time”,”direction”:”DESC”}]'
        }
    }
}
});
```

效果图：

	500mg/L含氯消毒液							
	台面				地面			
	日期	上午	下午	消毒人员	日期	上午	下午	消毒人员
☐	2022-06-08	已消毒	已消毒	超级管理员	2022-06-08	已消毒	已消毒	超级管理员
☐	2022-06-07	未消毒	已消毒	超级管理员	2022-06-07	已消毒	已消毒	超级管理员

form弹出层

- title 标题
- formLabelWidth 标签框宽度，默认80px
- items 表单元素，有select、input、textarea、cascader
- width:表单的宽度，高度是自适应的，默认是620

```
var vueObj = new Vue({
  el: '#app',
  data: function() {
          return {
            dialogs:[{
        ref: 'addDialog',
        formLabelWidth: 100px,
        width: 680, //不建议修改,正常一个系统统一一个宽度
        type:'',//默认值是dialog,如果写tableDialog就是打开一个表格
        title: '新增',
        items: [{},//多个元素
                    ]
            }]
    }
  }
});
```

常用案例

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>demo01</title>
    <link rel="stylesheet" href="../../lib/element-ui/element.css">
    <script src="../../common/constant.js"></script>
    <script src="../../common/utils.js"></script>
    <script src="../../lib/vue/vue.js"></script>
    <script src="../../lib/element-ui/element.js"></script>
    <script src="../../lib/jquery/jquery3.5.1.js"></script>
    <script src="../../component/base/base.js"></script>
    <link rel="stylesheet" href="../../component/table/table.css">
    <script src="../../component/table/table.js"></script>
    <script src="../../component/formDialog/formDialog.js"></script>
    <link rel="stylesheet" href="../../component/formDialog/formDialog.css">
</head>
<body>
<div id="app">
    <base-component ref="base"
                    @current-row-change="currentRowChange"
                    @query-param="queryParam"
                    :dialogs="dialogs"
                    :table="table"
                    :form="form">


    </base-component>
</div>
<script>
```

```
var GLOBAL_ENUM = {
    SEX: {
        CONVERT_TEXT: 'sexText',
        CONVERT: {
            1: '男',
            2: '女',
        }
    }
}

var vueObj = new Vue({
    el: '#app',
    data: function() {

        return {
            currentRow: null,
            form: {
                //操作按钮
                buttons: [
                    {type: 'add', bindRef: 'addDialog', label: '无需选中弹出层'},
                    {type: 'edit', bindRef: 'editDialog', label: '选中并带数据弹出层'},
                    {type: 'delete', url: "http://localhost:12006/WS/api/deleteurl", fields: ['opId', 'title'], message:"弹出删除的
提示,有默认提示可以不填写哈", label: '按钮四'},
                    {type: 'notice', url: "http://localhost:12006/WS/api/noticeurl", fields: ['opId', 'title'], message:"弹出提示的
提示,默认是删除,最后还是要写提示", label: '通知后台'},
                    {type: 'delete', url: "http://localhost:12006/WS/api/deleteurl", fields: ['opId', 'title'], label: '按钮6', cancel(){
                            console.log("删除的取消提示")
                        }, click :function(){
                        console.log("执行之间的事件,执行完就会直接返回了")
                        }},
                ],
                uploadButtons: [
                    {
                        type: 'add',
                        label: '点击上传文件1',
                        name: 'test',//文件上传时对应的名称
                        action: 'http://localhost:12006/WS/api/uploadTest',//后台action
                        accept: '.xls,.xlsx',//允许上传的类型
                        size: 1,//文件大小(默认是1,目前仅为1)
                        required: true,
                        fields: ['opId', 'title']
                    },
                    {
                        type: 'add',
                        label: '点击上传文件2',
                        action: 'http://127.0.0.1:12002/iom/api/uploadTest',//后台action
                        accept: '.pdf',//允许上传的类型
                        size: 1,//文件大小(默认是1,目前仅为1)
                        required: true,
                        fields: ['opId']
                    }
                ],
                //查询条件
                inputs: [
                    {name: 'inputFieldName', placeholder: '我是输入框', value: '输入test'},
                ],
                radios: [
```

```
    {
    name: 'area',
    value: 'city',
    onChange: function (val, form) {
        if (val === 'city') {
            form.cityCode = '';
            form.type = 1;
        } else {
            form.cityCode = '3';
            form.type = 2;
        }
    },
    options: [{
        label: '地级市',
        value: 'city'
    }, {
        label: '区县',
        value: 'county'
    }]
},{
    //按机构统计
    name: 'type',
    value: 1,
    onChange: function (val, form) {
        form.type = val;
    },
    dynamic: function (currentRow, form) {
        return form.area === 'county'
    },
    options: [{
        label: '按地区',
        value: 2
    }, {
        label: '按机构',
        value: 3
    }]
    }
],
dates: [
    {
    name: '-daterange',
    type: 'daterange',
    startPlaceholder: '事件开始日期',
    endPlaceholder: '事件结束日期'
}
],
selects: [
    {
        name: 'adviceStatus',
        value: 0,
        placeholder: '收费状态',
        clearable: false,
        options: [{
            value: 0,
            label: '待收费'
        }, {
            value: 1,
            label: '已收费'
```

```
            },{
                value: 2,
                label: '作废'
            },{
                value: 3,
                label: '回退'
            }]
        },{
            //如果需要后台指定下拉框的值,这里的id和isRemoteSelected一定要写上
            //id不能重复谢谢
            name: 'selected',
            placeholder: '我是获取远程数据并选中的下拉框',
            optionsUrl: "./org3.json",
            id: 'testSelected',
            isRemoteSelected: true,
        }
    ]
},
table: {
    url: "tabledemodata.json",
    //url: "http://localhost:12006/WS/api/queryTest",
    // manual: true,
    // handler: function() {
    //     console.log("handler")
    // },
    dataProcess: function (datas) {
        return datas
    },
    cols: [
        {
            prop: "title",
            label: "标题",
            width: 100
        },
        {
            prop: "createTime",
            label: "创建时间",
        },
        {
            prop: "sexText",
            label: "性别"
        },{
            prop: "address",
            label: "地址"
        },{
            prop: "author",
            label: "作者"
        }
    ],
    enumMap: {
        sex: GLOBAL_ENUM.SEX
    },
    params: {
        'userName': 'yzl'
    },
    sort:'title'
},
dialogs:[{
```

```
ref: 'addDialog',
type:'',//默认值是dialog,如果写tableDialog就是打开一个表格
title: '新增',
items: [
    {
        label: '标题',
        type: 'input',
        value: '直接赋值',
        field: 'title',
        required: true,
        props: {
            placeholder: '请输入用户名'
        },
        rules: [{required: true, message: '请输入用户名', trigger: 'blur'}],
    },
    {
        label: '用户ID',
        type: 'input',
        field: 'userId',
        props: {
            placeholder: '用户登录时使用的ID'
        },
        rules: [{required: true, message: '请输入用户ID', trigger: 'blur'}],
    },
    {
        label: '密码',
        type: 'password',
        field: 'password',
        props: {
            placeholder: '请输入密码'
        },
        rules: [{required: true, message: '请输入密码', trigger: 'blur'}],
    },
    {
        label: '手机号',
        type: 'input',
        field: 'userPhone',
        props: {
            placeholder: '初始密码为手机号后六位'
        },
        rules: [{pattern: /^1[3456789]\d{9}$/, message: '请输入正确的手机', trigger: 'blur'}],
    },
    {
        label: '邮箱',
        type: 'input',
        field: 'userEmail',
        props: {
            placeholder: '请输入邮箱'
        },
        rules: [{type: 'email', message: '请输入正确的邮箱', trigger: 'blur'}],
    },
    {
        label: '下拉框',
        type: 'select',
        field: 'org',
        optionsUrl: "./orgdata.json",
        props: {
            filterable: true
```

```
            },
            rules: [{required: true, message: '请选择机构', trigger: 'blur'}],
        },{
            label: '只能输入数字',
            type: 'input',
            field: 'userName',
            props: {
                placeholder: '请输入缴费金额',
                maxlength: 3,
                oninput: "value=value.replace(/[^0-9.]/g,'')"
            },
            rules: [{required: true, message: '请输入用户名', trigger: 'blur'}],
        },{
            label: '选中值的下拉框',
            type: 'select',
            field: 'orgId2',
            optionsUrl: "./org2.json",
            isRemoteSelected: true,//后台指定选中值,格式为{"value":"v","datas":[{"label":"标签"},{"value":"v"}]}, 这里最后会选中value
            //needName: true,//默认为不存在该属性,当该属性存在时则会提交下拉框的name值,规则为field + Name,这里为orgIdName
            rules: [{required: true, message: '请选择', trigger: 'blur'}],
            props: {
                filterable: true
            },
        },
        {
            label: '时间1',
            type: 'time',
            field: 'datetime',
            props: {
                placeholder: '请选择时间',
                'value-format': 'yyyy-MM-dd',
                'type': 'datetime' //year/month/date/week/datetime/datetimerange/daterange
            },
            rules: [{ message: '请选择时间', trigger: 'blur'}],
        },
        {
            label: '时间2',
            type: 'time',
            field: 'year',
            props: {
                placeholder: '请选择时间',
                'value-format': 'yyyy-MM-dd',
                'type': 'year'//year/month/date/week/datetime/datetimerange/daterange
            },
            rules: [{message: '请选择时间', trigger: 'blur'}],
        },{
            label: '时间3',
            type: 'time',
            field: 'date',
            props: {
                placeholder: '请选择时间',
                'value-format': 'yyyy-MM-dd',
                'type': 'date'//year/month/date/week/datetime/datetimerange/daterange
            },
            rules: [{message: '请选择时间', trigger: 'blur'}],
        },{
```

```
            label: '时间4',
            type: 'timerange',
            timeFormat: 'yyyy-MM-dd',
            timeType: 'daterange', //monthrange,datetimerange,daterange
            timeKeys: ['startTime', 'endTime'],
            field: 'date5',
            rules: [{message: '请选择时间', trigger: 'blur', required:false}]
        },{
            label: '时间5',
            type: 'timerange',
            timeFormat: 'yyyy-MM',
            timeType: 'monthrange', //monthrange,datetimerange,daterange
            timeKeys: ['startTime3', 'endTime3'],
            field: 'date7',
            value: [new Date(2000, 10, 10, 10, 10), new Date(2000, 10, 11, 10, 10)],// ['2020-03', '2020-04']
            rules: [{message: '请选择时间', trigger: 'blur', required:true}]
        }
    ],
    submitUrl: "http://localhost:12006/WS/api/addForm",
    //manual: false,//设置为false就不用写下面的handler了
    //module: addHeaderImgMod,
    // handler: function (innerVm, formData, rawData, cb, ref, url) {
    //     //如果有配置这个并且manual为false在post之后还会执行这个函数
    //     console.log("demo1")
    //     console.log(formData);
    //     //console.log(innerVm, formData, rawData, cb, ref, url);
    //     console.log(formData.startTime);
    //     console.log(formData.endTime);
    //     console.log(formData.startTime3);
    //     console.log(formData.endTime3);
    //     console.log(formData.startTime2);
    //     console.log(formData.endTime2);
    //     console.log(url)
    //     innerVm.$message.error("配置manual改为主动提交至后台，这里演示无法提交")
    // }
},{
    ref: 'editDialog',
    title: '编辑',
    items: [
        {
            label: '标题',
            type: 'input',
            field: 'title',
            readonly: true,
            props: {
                placeholder: '请输入标题'
            },
            rules: [{required: true, message: '请输入用户名', trigger: 'blur'}],
        },
        {
            label: '创建时间',
            type: 'input',
            field: 'createTime',
            props: {
                placeholder: '请输入创建时间'
            },
        },
        {
```

```
                label: '性别',
                type: 'input',
                field: 'sex',
                props: {
                    placeholder: '请输入性别'
                },
                // rules: [{pattern: /^1[3456789]\d{9}$/, message: '请输入正确的手机', trigger: 'blur'}],
            },
            {
                label: '手机号',
                type: 'input',
                field: 'phone',
                props: {
                    placeholder: '正则匹配手机号'
                },
                rules: [{pattern: /^1[3456789]\d{9}$/, message: '请输入正确的手机', trigger: 'blur'}],
            },
            {
                label: '地址',
                type: 'input',
                field: 'address',
                props: {
                    placeholder: '请输入邮箱'
                },
            },
            {
                label: '下拉框',
                type: 'select',
                field: 'orgId',
                optionsUrl: "./orgdata.json",
                needName: true,//默认为不存在该属性,当该属性存在时则会提交下拉框的name值,规则为field + Name,这里为orgIdName
                rules: [{required: true, message: '请选择', trigger: 'blur'}],
                props: {
                    filterable: true
                },
            },
            {
                label: '数字值下拉框',
                type: 'select',
                field: 'numberSelect',
                optionsUrl: "./numberSelect.json",
                needName: true,//默认为不存在该属性,当该属性存在时则会提交下拉框的name值,规则为field + Name,这里为orgIdName
                rules: [{required: true, message: '请选择', trigger: 'blur'}],
                props: {
                    filterable: true
                },
            },{
                label: '选中值的下拉框',
                type: 'select',
                field: 'orgId2',
                optionsUrl: "./org2.json",
                isRemoteSelected: true,//后台指定选中值,格式为{"value":"v","datas":[{"label":"标签"},{"value":"v"}]}, 这里最后会选中value
                //needName: true,//默认为不存在该属性,当该属性存在时则会提交下拉框的name值,规则为field + Name,这里为orgIdName
                rules: [{required: true, message: '请选择', trigger: 'blur'}],
```

```
                props: {
                    filterable: true
                },
            },{
                label: '时间3',
                type: 'time',
                field: 'createTime3',
                props: {
                    placeholder: '请选择时间',
                    'value-format': 'yyyy-MM-dd',
                    'type': 'date'//year/month/date/week/datetime/datetimerange/daterange
                },
                rules: [{required: true, message: '请选择时间', trigger: 'blur'}],
            }
        ],
        paramFields: ['opId'],
        enumMap: {
            sex: GLOBAL_ENUM.SEX
        },
        requireRow: true,
        submitUrl: "http://localhost:12006/WS/api/addForm",
        // manual: true,
        // handler: function (innerVm, formData, rawData, cb, ref, url) {
        //     console.log("test edit handler")
        //     console.log(formData);
        //     console.log(innerVm, formData, rawData, cb, ref, url);
        //     _this.$message.error("配置manual改为主动提交至后台，这里演示无法提交")
        // }
    }],
  }
},
mounted: function (){
  //testSelected其实就是上面指定的id，切记不能重复
  //这里要回调base.js的函数才能绑定值,这是固定套路写法
  this.$refs.base.getSelectedCombox("testSelected")
},
methods: {
  currentRowChange: function (val) {
    this.currentRow = val
  },
  queryParam: function(val) {
    //console.log(val)
  },
  handleSelectChange:function (json) {

    console.log(json.field);
    console.log(json.value);

    var param = {
      'ref':'addDialog',
      'fields': {
        "title": "下拉框赋值改变标题",
        "userId": "下拉框改变userId"
      }
    }
    //方法一
    this.$refs.base.setDialogComponentValue(param)
    //方法二
```

```
                var formData = this.$refs.base.getDialogFormData('addDialog')
                formData['userPhone'] = 13799118590
            }
        }
    })

</script>
</body>
</html>
```

分割线

```
var vueObj = new Vue({
  el: '#app',
  data: function() {
          return {
            dialogs:[
        ref: 'addDialog',
        type:'',//默认值是dialog,如果写tableDialog就是打开一个表格
        title: '新增',
        items: [{
                type: 'seperator'
         },//{}
                    ]
      ]
    }
  }

});
```

弹出层含上传功能(同步)示例说明

要点一：如果要自己写上传的funtion, 要注意这个方法： `handler: function (innerVm, formData, rawData, cb, ref, url)` ，

文件在formData里面，要这样获取： `formData.get("fileKey")`

要点二：编辑的时候要注意 `uploader.props.name` 的值要和后台传输的值对应,例如后台图片保存的地址的字段是fileName，而 `uploader.props.name` 的值为mediaFiles则需要在SQL中写明 `fileName as mediaFile` ，然后这里是大小写敏感的，如果是postgresql则需要 `fileName as "mediaFile"` ，且后台回写前台的关键代码在show()中

要点三：编辑的时候传输到前台的url值默认是使用逗号(,)进行分割,如果需要个性化则在 `uploader.props.split` 中写明符号，建议使用,分割用封号(;)也可以，但是用别的符号报错概不负责哦

要点四：编辑的时候是有缩略图的，如果是mediaFile，则缩略图key为mediaFileThumbUrl，缩略图key可以不存在，

要点五：删除的时候后台会接收一个deleteFile的key的值，这个是以逗号(,)为分割的数据

```
var GLOBAL_OPTS = {
  EVENT_TYPE: [
    {label: '枚举值1', value: 1},
    {label: '枚举值2', value: 2}
  ],
};//这里定义一个全局的枚举值,下面有用

var vueObj = new Vue({
```

```
data:function() {
  //定义uploader组件相关信息
  var newUploaderMod = {
    uploader:
      {
        fileUrls: '', //文件url
        thumbUrls: '',//如果有缩略图的话有值
        label: '上传调查说明（支持图片、PDF、Word。仅供内部查阅，不对外公开）',
        accept: 'image/*,.pdf,.doc,.docx', //可以接受的文件类型
        size: 1,
        props: {
          name: 'mediaFiles', //传到后台的key,
          split: ';'//默认是,可以不写
        },
        list:true, //列表形式,该参数可以不填,不填就是默认的卡片模式,推荐使用列表,卡片有BUG,再回填的时候没办法赋上缩略图
      }
  };
  return {
    form: {
      //通常情况下弹出层都是一个按钮,点完弹出来,这里的bindRef就是指这个弹出层的id
      buttons: [
        {operation: 'view', bindRef: 'viewDialog', label: '查阅'}
      ],
      //查询条件
      inputs: [...省略不写],
      selects: [...省略不写],
      dates: [...省略不写],
    },
    dialogs: [
      {
        ref: 'viewDialog',//这个值绑定buttons中的bindRef
        title: '查阅',
        formLabelWidth: '85px',
        items: [
          {   //通常情况下我们都是需要选中table的一列数据,然后把这个数据带个弹出框架
            //这里的field就要和table组件的field对应上,这样就可以直接赋值了
            label: '普通框',
            type: 'input',
            field: 'eventTitle',
          },
          {
            label: '枚举值',
            type: 'select',
            field: 'eventType',
            options: GLOBAL_OPTS.EVENT_TYPE,//这个就是根据枚举显示值
          },
          {
            label: '文本域',
            type: 'textarea',
            field: 'description',
          },
        ],
        readonly: true,
        requireRow: true,//需要选中一行才能点击使用
        submitUrl: "xxx/xxx.do",
        manual: true, //手动提交,需要配合submitUrl和handler函数一起使用，毕竟表单提交不可能不要URL的吧?
        module: newUploaderMod, //弹出框的附加模块,这里就表示上传
        paramFields: ['opId'],//提交的时候把选中记录中的opId数据一起提交
```

```
        onShow: function(rawData) {
          console.log('onShow', rawData)
          //编辑要加上下面两段
          //editFileMod.uploader.fileUrls = rawData.fileAddress
          //editFileMod.uploader.fileNames = rawData.fileAddress

        },
        //manual为false就不用写handler了,会自动提交
        handler: function (innerVm, formData, rawData, cb, ref, url) {
          //这里就是manual:true时的回调函数,url其实就是submitUrl
          _this.$refs.actionForm.validate(function (valid, object) {
            console.log('validate', valid, object)
            if (!valid) {
              cb()
              return
            }
            $.ajax({
              method: "POST",
              url: _url,
              data: formData,
              processData: false,
              contentType: false,
            }).done(function (resData) {
              // var resData = JSON.parse(res)
              if (resData.code !== 100) {
                innerVm.$message.error(resData.desc || '操作失败');
                cb()
                return
              }
              innerVm.$message.success("操作成功");
              innerVm.refreshTable()
              innerVm.getComp(ref).hide()
              cb()
            }).fail(function (res) {
              console.error(res)
              innerVm.$message.error("网络异常");
              cb()
            })
          })
        },
      }
    ],
  }
 }
});
```

接下来给出关键解析

```
//首先要写一个这个uploader对象,最早的时候建伟是针对这个对象进行封装
//意思就是说在编辑的时候需要自行把fileUrls和thumbUrls参数填充现在最
//新的版本是不需要了根据key也就是下面的mediaFiles以及后台的参数配合,以下面的mediaFiles为例
//后台的参数配合是指缩略图的key是后面拼一个ThumbUrl，比如这里就说mediaFilesThumbUrl,
//现在也就是后台传回前端页面数据的文件url也用mediaFiles这个值,框架就会根据行数据中的mediaFiles
//对应的值进行解析,这样就不用多写了
var editUploaderMod = {
    uploader:
        {
            fileUrls: '', //文件url
```

```
            thumbUrls: '',//如果有缩略图的话有值
            label: '上传附件（支持图片、PDF、Word。仅供内部查阅，不对外公开）',
            accept: 'image/*,.pdf,.doc,.docx', //可以接受的文件类型
            size: 6,
            props: {
                split: ';',
                name: 'mediaFiles' //传到后台的key
            }
        }
};
//其次在dialogs中配置
//这里假设我们需要特殊的情况配置要手动写后台提交的代码
//常规情况这里不需要写后台提交的代码,也就是handler
dialogs:[{
    ref: 'addDialog',
    type:'',
    title: '发布科普活动',
    items: [

    ],
    module: editUploaderMod,
    submitUrl: GLOBAL_API.POP_ACTIVE.PUBLISH,
          manual: true,//manual为true时必须要写handler方法
    handler: function (innerVm, formData, rawData, cb, ref, url) {
        //按这种方式提交后台可以接收到文件参数
        _this.$refs.popActiveForm.validate(function (valid, object) {

            if (!valid) {
                cb()
                return
            }

            formData.append("title", _this.activeModel.title)
            formData.append("theme", _this.activeModel.theme)
            formData.append("content", _this.activeModel.content)

            $.ajax({
                method: "POST",
                url: url,
                data: formData,
                processData: false,
                contentType: false,

            }).done(function (resData) {

                if (resData.code !== 100) {
                    innerVm.$message.error(resData.desc || '操作失败');
                    cb()
                    return
                }
                innerVm.$message.success("操作成功");
                innerVm.refreshTable()
                innerVm.getComp(ref).hide()
                cb()
            }).fail(function (res) {
                console.error(res)
                innerVm.$message.error("网络异常");
                cb()
```

```
            })
        })
    }
}]
```

然后我们看一下后台代码,注意一下 `editPopActiveAction` 中的 `@RequestParam(required = false) String deleteFile` ，这个deleteFile是你在前端页面上删除了的文件的链接，同时 `editPopActiveAction` 中的 `MultipartFile[] mediaFiles` 参数只会出现新增的数据意思是说旧有的文件数据是不会上传的

```
/**
 * 发布科普活动
 * @return
 */
@PostMapping(value = "/popActiveAction/publishPopActiveAction", produces = "application/json; charset=utf-8")
public String publishPopActiveAction(HttpServletRequest request,
                                     MultipartFile[] mediaFiles,
                                     @RequestParam(required = true) String title,
                                     @RequestParam(required = true) String content,
                                     @RequestParam(required = true) String theme) {
    return _popActiveService.publishPopActiveAction(entity, SessionUtils.getUserSession(request), mediaFiles);
}

/**
 * 编辑科普活动
 * @return
 */
@PostMapping(value = "/popActiveAction/editPopActiveAction", produces = "application/json; charset=utf-8")
public String editPopActiveAction(HttpServletRequest request,
                                     MultipartFile[] mediaFiles,
                                     @RequestParam(required = true) String opId,
                                     @RequestParam(required = false) String deleteFile,
                                     @RequestParam(required = true) String title,
                                     @RequestParam(required = true) String content,
                                     @RequestParam(required = true) String theme) {
    return _popActiveService.editPopActiveAction(entity, mediaFiles, deleteFile);
}
```

最后我们再解释一下 `deleteFile` 这个参数是怎么用的:

```
String[] urls = deleteFile.split(",");//deleteFile的形式为: url1,url2,url3

for(String url: urls) {

    if(StrUtil.isNotEmpty(url)) {
        //在文件服务器中删除这个文件,这个文件的存储路径是：桶名字+主键+文件名
        //桶名字默认有了所以这里是主键+文件名
        _minioServiceUtil.deleteFile(popScienceActiveEntity.getOpId()+ "/" + FileUtil.getFileNameWithUrl(url));
        //这个annexAddress是旧的链接的集合(形式为url1,url2,url3....)
        //这里的意思就是从annexAddress中删除url
        annexAddress = annexAddress.replace(url + ";", "");
    }
}
```

弹出层上传功能(异步)+原生elementui代码示例

```
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>demo02</title>

    <link rel="stylesheet" href="../../lib/element-ui/element.css">
    <link rel="stylesheet" href="../../css/common.css">

    <script src="../../lib/vue/vue.js"></script>
    <script src="../../lib/element-ui/element.js"></script>
    <script src="../../lib/jquery/jquery-3.3.1.min.js"></script>
    <script src="../../lib/echarts/echarts.common.min.js"></script>

    <script src="../../component/base/base.js"></script>
    <link rel="stylesheet" href="../../component/table/table.css">
    <script src="../../component/table/table.js"></script>
    <link rel="stylesheet" href="../../component/formDialog/formDialog.css">
    <script src="../../component/formDialog/formDialog.js"></script>
    <script src="../../component/formDialog/uploadSync.js"></script>


    <script src="../../common/config.js"></script>
    <script src="../../common/utils.js"></script>
</head>
<body>
<div id="vue">
    <base-component ref="base" :form="form" :table="table" :dialogs="dialogs" @current-row-change="
currentRowChange">
        <template slot="uploadSyncSlot">
            <!--表单-->
            <el-form :inline="true" status-icon class="action-form" ref="actionForm" :rules="actionFormRules" :model="
actionFormData" size="mini">
                <!--查看流程-->
                <div>
                    <el-form-item label="当前流程" label-width="85px">
                        <el-button type="primary" size="mini" @click="showProgress">查看</el-button>
                    </el-form-item>
                </div>
                <div>
                    <el-form-item label="审批人" label-width="85px">
                        <el-input v-model="actionFormData.user" placeholder="审批人"></el-input>
                      </el-form-item>

                </div>
                <el-form-item label="待审核" label-width="85px">
                    <el-radio-group v-model="action">
                        <el-radio-button label="通过">通过</el-radio-button>
                        <el-radio-button label="退回">退回</el-radio-button>
                    </el-radio-group>
                </el-form-item>

                <div v-show="action === '通过'">
                    <el-form-item label="公示案件" label-width="85px">
                        <el-radio-group v-model="actionFormData.isPublic">
                            <el-radio-button :label="1">是</el-radio-button>
                            <el-radio-button :label="0">否</el-radio-button>
                        </el-radio-group>
```

```
            </el-form-item>
        </div>
        <!--退回模块-->
        <div v-show="action === '退回'">
            <el-form-item prop="reply" label="反馈" label-width="85px">
                <el-input
                        type="textarea"
                        style="width: 446px"
                        :autosize="{ minRows: 2, maxRows: 4}"
                        placeholder="请输入反馈"
                        v-model.trim="actionFormData.reply">
                </el-input>
            </el-form-item>
        </div>
    </el-form>
    <label
        style="padding-left: 85px; margin-bottom: 10px;text-align: right;display: inline-block;padding: 0 12px 4px 0;
box-sizing: border-box;">上传组件</label>
    <el-card class="box-card">
        <el-upload :data="{opId:opId}"
        action="http://127.0.0.1:12005/AMP/operationsManageAction/test"
        list-type="picture-card"
        :file-list="myFileList"
        :on-success="handleSuccess"
            :auto-upload="true">
            <i class="el-icon-plus" style="margin-top:35px"></i>
            <div slot="file" slot-scope="{file}">
                <img class="el-upload-list__item-thumbnail" :src="file.url" alt="">
                <span style="text-align: left" class="el-upload-list__item-actions">
                    <span class="el-upload-list__item-preview" @click="handlePictureCardPreview(file)">
                        <i style="padding-left: 10px;width: 5px;height: 5px;" class="el-icon-zoom-in"></i>
                    </span>
                    <span v-if="!disabled" class="el-upload-list__item-delete" @click="handleDownload(file)">
                        <i style="width: 5px;height: 5px;" class="el-icon-download"></i>
                    </span>
                    <span v-if="!disabled" class="el-upload-list__item-delete" @click="handleRemove(file)">
                        <i  style="width: 5px;height: 5px;" class="el-icon-delete"></i>
                    </span>
                </span>
            </div>
        </el-upload>
        <!-- <el-dialog :visible.sync="dialogVisible">
            <img width="100%" :src="dialogImageUrl" alt="">
          </el-dialog> -->
    </el-card>
  </template>
 </base-component>
</div>
<script>

 var GLOBAL_ENUM = {
    SEX: {
        1: '男',
        2: '女'
    }
 }
```

```
var vueObj = new Vue({
    el: '#vue',
    created: function() {
        this.opId = this.generateUUID();
    },
    data: function () {
        var _this = this
        var newUploaderMod = {
        uploader:
            {
                fileUrls: '', //文件url
                thumbUrls: '',//如果有缩略图的话有值
                label: '上传说明（支持图片、PDF、Word、mp4。仅供内部查阅，不对外公开）',
                accept: 'image/*,.pdf,.doc,.docx,.mp4', //可以接受的文件类型
                props: {
                        name: 'mediaFiles', //传到后台的key
                }
            }
        };
        return {
            myFileList:[],
            opId: 'ok',
            currentRow: null,
            disabled: false,
            action: '通过',
            actionFormData: {
                reply: '',
                user:'',
                isPublic: 0
            },
            actionFormRules: {
                reply : [{required: true, message: '请输入反馈', trigger: 'change'}],
            },
            //图片预览
            dialogVisible: false,
            dialogImageUrl: '',
            form: {
                //操作按钮
                buttons: [
                    {operation: 'add', bindRef: 'uploadDialog', label: '同步上传功能组件'},
                    {bindRef: 'uploadSync', operation: 'add', label: '原生+异步上传功能组件'}
                ],
                uploadButton:{
                    operation: 'add',
                    label: '点击上传文件1',
                    action: 'http://127.0.0.1:12002/iom/api/uploadTest',//后台action
                    accept: '.xls,.xlsx',//允许上传的类型
                    size: 1,//文件大小(默认是1,目前仅为1)
                },
                //查询条件
                inputs: [
                    {name: 'inputFieldName', placeholder: '我是输入框'},
                ],
                dates: [{
                        name: '-daterange',
                        type: 'daterange',
                        startPlaceholder: '事件开始日期',
                        endPlaceholder: '事件结束日期'
```

```
        }, ],
    selects: [
        {name: 'selectFieldName', placeholder: '我是下拉框', options: [{
                value: 1,
                label: '选项1'
            }, {
                value: 2,
                label: '选项2'
            }]
        }, {
            name: 'selectdataFieldName',
            placeholder: '我是获取远程数据下拉框',
            optionsUrl: "./orgdata.json"
        }
    ],
},
table: {
    url: "tabledemodata.json",
    cols: [
        {
            prop: "title",
            label: "标题",
            width: 100
        },
        {
            prop: "createTime",
            label: "创建时间",
        },
        {
            prop: "sex",
            label: "性别"
        },{
            prop: "address",
            label: "地址"
        },{
            prop: "author",
            label: "作者"
        }
    ],
    enumMap: {
        sex: GLOBAL_ENUM.SEX
    },
    params: {}
    // sort: '[{"property":"create_time","direction":"DESC"}]'
},
dialogs:[{
    ref: 'uploadSync',
    title: '异步上传组件',
    formLabelWidth: '85px',
    submitUrl: "http://127.0.0.1:12005/AMP/operationsManageAction/test",
    items: [],
    manual: true,
    onShow: function(rawData) {
        console.log('onShow', rawData)
        _this.actionFormData.reply = '初始化原生模型'
        // _this.picUrls = rawData.imageUrl ? rawData.imageUrl.split(';') : []
        // _this.picUrls.pop()
        // _this.voiceUrl = rawData.speechUrl ? rawData.speechUrl.slice(0, -1) : '' //去除末尾；
```

```
    // _this.videoUrl = rawData.videoUrl ? rawData.videoUrl.slice(0, -1) : ''
    //执法图片
    // _this.lawPicUrls = rawData.lawAnnexes ? rawData.lawAnnexes.split(',') : []
    // _this.lawPicUrls.pop()
    // _this.lawThumbPicUrls = rawData.lawThumbImage ? rawData.lawThumbImage.split(',') : []
    // _this.lawThumbPicUrls.pop()
    // uploaderMod.uploader.fileUrls = rawData.lawAnnexes ? rawData.lawAnnexes : ''
    // uploaderMod.uploader.fileNames = rawData.lawFileName ? rawData.lawFileName : ''
},
handler: function (innerVm, formData, rawData, cb, ref, url) {
        // console.log(formData, rawData, cb, ref, url)
        var _url = url //默认为通过的url
        var needValidate = false
        var params = {
            opId: _this.opId,
            user: _this.actionFormData.user
        }

        switch (_this.action) {
            case '通过':
                params.isPublic = _this.actionFormData.isPublic;

                break
            case '退回':
                //可以设置url
                params.reply = _this.actionFormData.reply
                needValidate = true
                break
        }

        if(needValidate) {
            if(_this.actionFormData.reply == null || _this.actionFormData.reply === '') {
                innerVm.$message.warning('请输入反馈');
                cb()
                return
            }
        }
        // console.log(_this.actionFormData)
        $.ajax({
            method: "GET",
            url: _url,
            data: params,
        }).done(function (resData) {
            // var resData = JSON.parse(res)
            if (resData.code !== 100) {
                innerVm.$message.error(resData.desc || '操作失败');
                cb()
                return
            }
            innerVm.$message.success("操作成功");
            innerVm.refreshTable()
            innerVm.getComp(ref).hide()
            cb()
        }).fail(function (res) {
            innerVm.$message.error("网络异常");
            cb()
        })
    },
```

```
},{
    ref: 'uploadDialog',
    title: '上传组件',
    formLabelWidth: '85px',
    items: [
        {   //通常情况下我们都是需要选中table的一列数据,然后把这个数据带个弹出框架
        //这里的field就要和table组件的field对应上,这样就可以直接赋值了
            label: '普通框',
            type: 'input',
            field: 'eventTitle',
        },{
            label: '下拉框',
            type: 'select',
            field: 'orgId',
                optionsUrl: "./orgdata.json",
                rules: [{message: '请选择', trigger: 'blur'}],
            props: {
                filterable: true
            },
        },{
            label: '文本域',
            type: 'textarea',
            field: 'description',
        },
    ],
    //requireRow: true,//需要选中一行才能点击使用
    submitUrl: "http://127.0.0.1:12005/AMP/operationsManageAction/test",
    manual: false, //手动提交,需要配合submitUrl和handler函数一起使用，毕竟表单提交不可能不要URL的吧?
    module: newUploaderMod, //弹出框的附加模块,这里就表示上传
    onShow: function(rawData) {
        console.log('onShow', rawData)
    },
    handler: function (innerVm, formData, rawData, cb, ref, url) {
        //这里就是manual:true时的回调函数,url其实就是submitUrl
        _this.$refs.actionForm.validate(function (valid, object) {
            console.log('validate', valid, object)
            if (!valid) {
                cb()
                return
            }
            $.ajax({
                method: "POST",
                url: _url,
                data: formData,
                processData: false,
                contentType: false,
            }).done(function (resData) {
                // var resData = JSON.parse(res)
                if (resData.code !== 100) {
                    innerVm.$message.error(resData.desc || '操作失败');
                    cb()
                    return
                }
                innerVm.$message.success("操作成功");
                innerVm.refreshTable()
                innerVm.getComp(ref).hide()
                cb()
            }).fail(function (res) {
```

```
                                console.error(res)
                                innerVm.$message.error("网络异常");
                                cb()
                            })
                        })
                    },
                }
            ],
        }
    },
    methods: {
        currentRowChange: function (val) {
            this.currentRow = val
        },
        handleSuccess(response, file, fileList) {
            this.myFileList.push(file)
        },
        handleRemove(file) {
            //这里要通知后台删除这个文件
            console.log(file);
            console.log(this.myFileList)
            this.myFileList=[]
        },
        handlePictureCardPreview(file) {
            //这两个参数要配合upload下面被注释的代码使用,其实就是在upload下方显示这个图片
            this.dialogImageUrl = file.url;
            this.dialogVisible = true;
            window.open(file.url)
        },
        handleDownload(file) {
            //去后台下载
            console.log(file);
        },
        showProgress: function () {
            //this.$refs.base.getComp('progressDialog').show()
        },
        generateUUID: function () {
            var d = new Date().getTime();
            if (window.performance && typeof window.performance.now === "function") {
                d += performance.now(); //use high-precision timer if available
            }
            var uuid = 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function (c) {
                var r = (d + Math.random() * 16) % 16 | 0;
                d = Math.floor(d / 16);
                return (c == 'x' ? r : (r & 0x3 | 0x8)).toString(16);
            });

            var uuid = uuid.replace(/-/g, "");

            return uuid;
        }
    },
})
</script>
</body>
</html>
```

普通弹出层(含隐藏字段上传)

```
{
  ref: 'editDialog',
    title: '编辑',
      items: [
        {
          label: '用户名',
          type: 'input',
          field: 'userName',
          props: {
            placeholder: '请输入用户名'
          },
          rules: [{required: true, message: '请输入用户名', trigger: 'blur'}],
        },
        {
          label: '用户ID',
          type: 'input',
          field: 'userId',
          props: {
            placeholder: '用户登录时使用的ID'
          },
          rules: [{required: true, message: '请输入用户ID', trigger: 'blur'}],
        },
        {
          label: '密码',
          type: 'password',
          field: 'password',
          props: {
            placeholder: '请输入密码'
          },
          rules: [{required: true, pattern: /^.{0,15}$/, message: '请输入小于8位的密码', trigger: 'blur'}],
        },{
          label: '性别',
          type: 'select',
          field: 'sex',
          options:[{"label":"男", "value":1},{"label":"女", "value":2}],
          props: {
            filterable: true
          },
          rules: [{required: true, message: '请选择性别', trigger: 'blur'}],
        },{
          label: '家庭地址',
          type: 'input',
          field: 'userAddress',
          props: {
            placeholder: '请输入地址'
          },
          rules: [{trigger: 'blur'}],

        },
        {
          label: '手机号',
          type: 'input',
          field: 'userPhone',
          props: {
            placeholder: '初始密码为手机号后六位'
```

```
        },
        rules: [{pattern: /^1[3456789]\d{9}$/, message: '请输入正确的手机', trigger: 'blur'}],
      },
      {
        label: '邮箱',
        type: 'input',
        field: 'userEmail',
        props: {
          placeholder: '请输入邮箱'
        },
        rules: [{type: 'email', message: '请输入正确的邮箱', trigger: 'blur'}],
      },
      {
        label: '机构',
        type: 'select',
        field: 'orgId',
        optionsUrl: GLOBAL_API.OPTS.OGR_COMMBOX,
        props: {
          filterable: true
        },
        rules: [{required: true, message: '请选择机构', trigger: 'blur'}],
      },
      {
        label: '资料权限',
        type: 'select',
        field: 'dmPer',
        options:[{"label":"全部", "value":1},{"label":"只读", "value":2}],
        props: {
          filterable: true
        },
        rules: [{required: true, message: '请选择机构', trigger: 'blur'}],
      }, {
        label: '出生年月',
        type: 'time',
        field: 'birthday',
        props: {
          filterable: true,
          'value-format': 'yyyy-MM-dd',
          'type': 'date'
        },
        rules: [{message: '请选择出生年月', trigger: 'blur'}],
      }
    ],
    paramFields: ['opId'],
    submitUrl: GLOBAL_API.USER.EDIT,
    requireRow: true
},
```

form表单下拉框和顶部条件查询下拉框

格式为 `{"value":"选中这个值1","datas":[{"name":"标签1"},{"value":"选中这个值1"},{"name":"标签2"},{"value":"值2"}]}`

form表单的定义下拉框在return的dialogs，即:

```
var vueObj = new Vue({
  el: '#vue',
```

```
        data: function () {
                ...//省略代码
            return {
                ...//省略代码
                    dialogs:[{
                label: '选中值的下拉框',
                type: 'select',
                field: 'orgId2',
                optionsUrl: "./org2.json",
                isRemoteSelected: true,//后台指定选中值,格式为{"value":"v","datas":[{"name":"标签"},{"value":"v"}]}，这里最后会选中value
                //needName: true,//默认为不存在该属性,当该属性存在时则会提交下拉框的name值,规则为field + Name,这里为orgIdName
                rules: [{required: true, message: '请选择', trigger: 'blur'}],
                props: {
                    filterable: true
                    },
              },{
                field: 'remoteInputSelect',
                label: '远程文字下拉框',
                type: 'select',
                placeholder: '输入下拉框',
                options: [],
                props: {
                  filterable: true,
                  remote: true,
                  //multiple: true, //不支持多个查询
                  'reserve-keyword': true,
                  'remote-method': function(query) {
                    _this.dialogs[0].items[1].options = [{label: 'tt', value: '111'},{label: 'tt2', value: '222'}]
                  }
                },
              },
            ]
          }
        }
      })
```

form下拉框及change事件

```
{
  label: '所属机构',
    type: 'select',
    field: 'org',
    optionsUrl: GLOBAL_API.ORG.COMBOX,
    props: {
    filterable: true
  },
  rules: [{required: true, message: '请选择机构', trigger: 'blur'}]
}

{
    label: '性别',
    type: 'select',
    field: 'sex',
    props: {
        filterable: true,
```

```
        options: [{
            label: '男',
            value: '1'
        },{
            label: '女',
            value: '2'
        }],
    },
    rules: [{message: '请选择性别', trigger: 'blur'}],
}
```

在框架一开始的 `<base-component>` 中添加 `@combox-select-change` ，注意 `@combox-select-change` 不可以修改，为固定key,后面的 `handleSelectChange` 可以根据个人喜好修改

```
    <base-component ref="base"
                    :form="form"
                    :table="table"
                    :dialogs="dialogs"
                    @current-row-change="currentRowChange"
                    @combox-select-change="handleSelectChange">
      </base-component>
```

`handleSelectChange` 要添加到method中,比如你写一个demo.html，这个函数就在demo.html中定义:

```
new Vue({
        el: '#vue',
    data: function() {},
    methods: {
        //这样就获取了变化,field是用户自定的
        handleSelectChange: function (json) {
                console.log(json.field);
                console.log(json.value);
        }
    }
});
```

解析:formDialog.js中有一段代码,用来监听下拉框的变化,这里就是调用父框架的 `itemEvent` 方法:

```
handleSelectChange(field, value) {
        this.$emit('itemEvent', 'change', field, value)
},
```

父方法中的itemEvent实际调用的是onItemEvent，在onItemEvent中又调用了父组件的 `combox-select-change` ,所以上面一开始需要自定这个!

```
onItemEvent: function (arguments, fn) {
    var key = arguments[0];
    var json = {
        'field': arguments[1],
        'value': arguments[2]
        }
    if(key === 'change') {
        this.$emit("combox-select-change", json)
        }

    if (typeof fn === 'function') {
```

```
        fn.apply(null, arguments)
        }
},
```

为弹出层附固定值&下拉框change后对表单中的组件赋值

其实就是为formdata新增数据!

```
handleSelectChange:function (json) {
        //可以根据field进行判断需要改变什么指
    console.log(json.field);
    console.log(json.value);

    var param = {
        'ref':'addDialog', //这个就是form.dialog.ref的值
        'fields': {
            "title": "下拉框赋值改变标题", //title就是说有一个组件的field的值title需要重新赋值
            "userId": "下拉框改变userId" //大意同上
            }
    }

    this.$refs.base.setDialogComponentValue(param)
   }
```

解析:这里其实调用base.js中的 `setDialogComponentValue` ，然后再调用formDialog.js中的 `setComponentValue` ,再在这里面找到formData对其进行赋值,我试过双向绑定但是不行，因为框架是将模型绑定为里面定义好的formData，所以这里就引发出第二种解决方案，为什么不把formData返回去呢?

```
handleSelectChange:function (json) {

  console.log(json.field);
  console.log(json.value);

  var param = {
    'ref':'addDialog',
    'fields': {
      "title": "下拉框赋值改变标题",
      "userId": "下拉框改变userId"
    }
  }

  this.$refs.base.setDialogComponentValue(param)
  var formData = this.$refs.base.getDialogFormData('addDialog')
  formData['userPhone'] = 13799118590
   }
```

在tab页打开一个新的tab页面

在index.html有如下代码,这段是调用子组件 `framecomponent` 的 `openTaskWindow` 方法,它会把json参数直接带到frame.js对应的方法,这里注意type要为1,1是打开一个新窗口

```
window.addEventListener('message', function (e) {

    var param = e.data;
```

```
    switch (param.type) {
        //1就是打开一个windows
        case 1:
            vm.$refs.framecomponent.openTaskWindow(param)
            break;
        default:
            break;
    }
})
```

比如我们期望在页面A启用一个方法然后能打开一个新的页面B,那么我们需要在页面A写的点击方法写如下代码中的 `openTaskWindow` ,这样就可以调用到上面的父组件的方法,前面四个参数title、url、callUrl、type是必填:
title:tab页的标签
url:根据这个url去获取页面，url和callurl只能填写一个，如果都写则以callurl为准
callUrl:根据这个callUrl再结合其余的参数从后台拿回两个参数一个是title,一个是url,title就是tab页名字,url就是跟上一个参数url一个意思,格式:

其他参数: 其他参数是自定义，通常是结合callUrl在后台判断所用

```
[{"code":100,"datas":[{"title":"tab名字", "url":""}]}]
```

type: 1:打开一个window
其余参数:配合callUrl使用,会把参数带给后台

```
openTaskWindow: function(event) {

    click: function() {
        var param = {
            title: "测试标题",
            url: "/BNWeb/configMenuAction/unsatisfactory.do",
            callUrl:"",
            type: 1,
            nodeKey:"temp",
        };
        let targetOrigin = "*";
        window.parent.postMessage(param, targetOrigin);
                }
}
```

callUrl配合其他参数在后台使用：

```
            openTaskWindow: function(event) {
                //子组件调用父组件并弹出一个新的tab页
                var param = {
                    title: event.target.dataset.title,
                    nodeName: event.target.dataset.nodename,
                    opId: event.target.dataset.opid,
                    nodeKey: event.target.dataset.nodekey,
                    callUrl:"/BNWeb/frameAction/getFlowTitleAndUrl.do",
                    type: 1
                };
                let targetOrigin="*";
                window.parent.postMessage(param, targetOrigin);
            }
```

弹出的时候触发一个事件

```
dialogs:[{
  ref: 'addDialog',
  type:'',//默认值是dialog,如果写tableDialog就是打开一个表格
  title: '新增',
  event: function () {
        alert("testH event") //创建一个事件
  },
  items: [
      ]
},{}//...]
```

弹出层table

具体可以参考demo02.html。

先看一下参考数据，比方说我们返回的数据格式如下，我们在datas的第二个数据中新增一个table的数组,就是Key为table，当然这个table是可以任意替换的，显然我们的写法就是选中第二个数据的时候弹出能显示一个表格数据。

```
{
    "total": 2,
    "code": 100,
    "success": true,
    "datas": [{
        "opId": "test1",
        "title": "标题",
        "createTime": "2021-02-21",
        "sex": 1,
        "address": "福建省福州市",
        "author": "互医健康",
        "phone": "13799118590",
        "numberSelect": "1"
    }, {
        "opId": "test2",
        "title": "标题",
        "createTime": "2021-02-21",
        "sex": 1,
        "address": "福建省福州市",
        "author": "互医健康",
        "phone": "13799118590",
        "numberSelect": "2",
        "table":[{
            "opId": "test3",
            "title": "标题3",
            "createTime": "2021-02-21",
            "sex": 1,
            "address": "福建省福州市",
            "author": "互医健康",
            "phone": "13799118590",
            "numberSelect": "1"
        },{
            "opId": "test4",
            "title": "标题4",
            "createTime": "2021-02-21",
            "sex": 1,
            "address": "福建省福州市",
            "author": "互医健康",
```

```
                    "phone": "88888888",
                    "numberSelect": "1"
                }]
        }],
        "desc": "成功"
}
```

接下来我们新增一个弹出层，一定要添加一个event事件，具体参数看代码的注释

```
dialogs:[{
  ref: 'editDialog',
  title: '弹出表格',
  requireRow: true,
  width:800,
  event: function () {
    //第一个参数editDialog，就是填你弹出层的ref
    //第二个参数table就是你指定的数组的key，名字可以任何替换，要喝返回数据对应即可
    _this.$refs.base.setFormDialogTableData("editDialog", "table")
  },
  items: [
    //固定写type
    type: 'table',
    width: '700px',
    //注意:假如要自己定义width,这里width最好要根据实际列数进行调整，如果多了请调整dialogs的width默认值620，我调过800
    //当然默认的width是自适应的可以不写
    //prop就是对应后台返回数据的key
    cols: [{
          prop: "title",
          label: "标题",
              width: 100 //默认可以不写
        },
        {
          prop: "createTime",
          label: "创建时间",
          width: 100 //默认可以不写
        },
        {
          prop: "sexText",
          label: "性别",
          width: 100 //默认可以不写
        },{
          prop: "address",
          label: "地址",
          width: 100 //默认可以不写
        },{
          prop: "author",
          label: "作者",
          width: 100 //默认可以不写
        }
        ],
        ]
}]
```

form dialog的onClose事件

```
dialogs:[
  {
    ref: 'addDialog',
    type:'',
    title: '新增',
    items: [],
    submitUrl: "/action.do",
    manual: false,
    onClose:function(){
      console.log("关闭了")
    },
    handler: function (innerVm, formData, rawData, cb, ref, url) {}
  },
]
```

`onClose` 的作用是监听弹出窗的关闭事件，在该方法内可以在弹窗关闭的时候，执行必要的操作。

举个例子：由于一些特殊需求，使用了框架的弹窗，但是需要在弹窗中自定义页面，如下所示，这时候使用了表单组件以及多个el-upload组件。此时如果没有在弹窗关闭的时候置空表单或者归零数据，就有可能会使用旧数据，而导致提交上去的数据出现错误。值得注意的是：el-upload使用完之后需要及时的调用 this.$refs.id.clearFiles()方法来清空已上传的文件列表，否则之前的图片还会出现在新的弹窗中。

```
<div id="vue">
    <base-component ref="base" :form="form" :table="table" :dialogs="dialogs" @current-row-change="
currentRowChange">
        <template slot="addDialogSlot">
            <el-form  status-icon size="mini" label-width="80px" >
                <el-form-item label="选择年份" required>
                                    <el-select :disabled="true" size="mini" v-model="actionFormData.yearValue" placeholder="请
选择年份">
                                        <el-option v-for="item in actionFormData.years" :key="item.value" :label="item.label"
                                            :value="item.value">
                                        </el-option>
                                    </el-select>
                            </el-form-item>
                <el-form-item label="处置间场所环境" required>
                    <el-upload ref="viewDisposalRoomRef" action="#" multiple accept="image/*" list-type="picture-card"
                            :limit=9 :before-upload="beforeUploadPicture" :on-preview="handlePictureCardPreview"
                            :on-remove="handleDisposalRoomRemove" :on-progress="uploadProgress" :on-success="
uploadSuccess"
                            :on-error="uploadError" :show-file-list="true" :auto-upload="true"
                            :file-list="actionFormData.disposalRoomList" :disabled="true"
                            :http-request="disposalRoomHttpRequest">
                        <i class="el-icon-plus" style="line-height: inherit;"></i>
                    </el-upload>
                </el-form-item>
                <el-form-item label="垃圾桶垃圾袋封闭情况" required>
                    <el-upload ref="viewGarbageConditionRef" action="#" multiple accept="image/*" list-type="picture-card"
                            :limit=9 :before-upload="beforeUploadPicture" :on-preview="handlePictureCardPreview"
```

```
                    :on-remove="handleGarbageConditionRemove" :on-progress="uploadProgress"
                    :on-success="uploadSuccess" :on-error="uploadError" :show-file-list="true" :auto-upload="true"
                    :file-list="actionFormData.garbageConditionList" :disabled="true"
                    :http-request="garbageConditionHttpRequest">
                <i class="el-icon-plus" style="line-height: inherit;"></i>
              </el-upload>
            </el-form-item>
          </el-form>
        </template>
    </base-component>
</div>
```

源码分析：

`el-dialog` 组件本身自带关闭事件 `closed` ，这是Dialog关闭动画结束时的回调。所以只需要在 `formDialog.js` 文件中定义这个 `closed` 方法：

```
<el-dialog :title="title" :visible.sync="formVisible" @closed="onDialogClosed" class="form-dialog-comp" :width="width">
```

接着在 `onDialogClosed` 方法中使用 `$emit` 让 `base.js` 父组件可以监听到这个自定义事件：

```
onDialogClosed: function () {
  ...
  setTimeout(function (_this) {
    _this.$emit('close')
  }, 0, this)
}
```

最后在 `base.js` 文件的页面模板中找到dialog的模块，加上自定义的 `close` 事件， `close` 事件携带一个 `onClose` 属性，通过 `call` 将这个 `onClose` 属性集成到dialog的属性中：

```
<template v-for="item in dialogs">
      <form-dialog-component v-if="item.requireRow"
                             :ref="item.ref"
                             :width="item.width"
                             :readonly="item.readonly"
                             :no-footer="!item.submitUrl"
                             :title="item.title"
                             :items="item.items"
                             :event="item.event"
                             :module="item.module"
                             :current-row="currentRow"
                             :form-label-width="item.formLabelWidth"
                             @itemEvent="onItemEvent(arguments, item.onItemEvent)"
                             @show="onDialogShow(arguments, item.onShow)"
                             @close="onDialogClose(item.onClose)"
                             @valid="onDialogValid(arguments, item.ref, item.submitUrl, item.handler, item.manual, item.
```

```
params, item.paramFields)”
                                >
            <template slot=”middle”>
                <slot :name=”item.ref + \’Slot\’”></slot>
            </template>
        </form-dialog-component>
        <form-dialog-component v-else :ref=”item.ref”
                                :width=”item.width”
                                :readonly=”item.readonly”
                                :no-footer=”!item.submitUrl”
                                :title=”item.title”
                                :items=”item.items”
                                :event=”item.event”
                                :module=”item.module”
                                :subForm=”item.subForm”
                                :form-label-width=”item.formLabelWidth”
                                @itemEvent=”onItemEvent(arguments, item.onItemEvent)”
                                @show=”onDialogShow(arguments, item.onShow)”
                                @close=”onDialogClose(item.onClose)”
                                @valid=”onDialogValid(arguments, item.ref, item.submitUrl, item.handler, item.manual, item.
params, item.paramFields)”
                               >
            <template slot=”middle”>
                <slot :name=”item.ref + \’Slot\’”></slot>
            </template>
        </form-dialog-component>
    </template>



onDialogClose: function (fn) {
  if (typeof fn === ’function’) {
    fn.call(this)
  }
},
```

多选下拉框

注意默认赋值的话value一定要是数组类型

```
{
    label: ’多选下拉框’,
    type: ’multiSelect’,
    field: ’orgMulti’,
    //needName: true, //默认不写这个属性,如果为true的话是orgMultiName作为Key,且是一个数组
    optionsUrl: ”./orgdataMuliti.json”,
    props: {
        filterable: true
        },
    rules: [{required: true, message: ’请选择机构’, trigger: ’blur’}],
    value:[’v1’, ’v2’]
}
```

动态表格

动态表格集成在对话框中主要是用来新增或者展示多条结构类似的数据，写法参考了框架表格的实现，所以写法大体相似，如下所示：

```
{
        ref: 'addDialog',
        title: '带动态表格新增',
        width: '1100px', //对话框的宽度，一般要比底下的动态表格设置的宽度大
        items: [{
                label: '下拉框',
                type: 'input',
                field: 'orgId',
                rules: [{
                        message: '请选择',
                        trigger: 'blur'
                        }],
                props: {
                        filterable: true
                        },
        }, {
                type: 'dynamicTable', //动态表格的类型
                field: 'tableField', //key
                width: '1000px',
                cols: [{
                        prop: "title",
                        label: "标题",
                        type: 'textarea', //多行文本
                },
                        {
                                prop: "createTime",
                                label: "创建时间",
                                type: 'input', //输入框
                                //设置属性
                                props: {
                                        filterable: true,
                                        placeholder: '请选择科室'
                                        },
                                //验证规则
                                rules: [{
                                        message: '请选择科室',
                                        trigger: 'blur',
                                        required: false,
                                }],
                        },
                        {
                                prop: "gender",
                                label: "性别",
                                type: 'select', //下拉框
                                optionsUrl: "./orgdata.json", //远程数据接口地址
                        }, {
                                prop: "phone",
                                label: "电话",
                                type: 'time', //时间选择
                                props: {
                                        placeholder: '请选择时间',
                                        'value-format': 'HH:mm',
                                        format: "HH:mm",
```

```
                    },
                    rules: [{
                        message: '请选择时间',
                        trigger: 'blur'
                        }],
                },
                {
                    prop: "files",
                    label: "文件",
                    type: 'upload', //文件类型
                }
                ],
        }, ],
        submitUrl: "http://127.0.0.1:12002/iom/memberManagementAction/test",
        manual: false,
    },
```

可以看到，动态表格也是写在items的数组里面，跟以往的输入框是同一级别的，只不过item的类型此时是 `dynamicTable` ， `field` 是提交到后台的key，后台收到的是一个字符串类型的json数据，数据格式为 [{...},{...},{...}.....]。每个大括号" { } "代表表格里面的一整行数据，例如：动态表格里面有五行，提交的数组数据里面也会有五个大括号。

`cols` 里面声明的是表格的各列的数据配置，具体属性如下表所示：

属性名	说明
prop	所定义列的key，同时也是提交后台的字段名
label	显示的标题
type	类型,有input,textarea,select,switch,date,time,upload
props	组件的属性，例如空白时的提示，数据的格式化等等
rules	验证规则

这里有个需要注意的点，当 `col` 的类型为 `upload` 时，做新增操作的时候，前端会把 `upload` 组件的 `prop` 作为传递给后端的 `key` ，如下所示：

```
public String test(
  @RequestParam(required = false, name = "tableField") String tableField,
  @RequestParam(required = false, name = "files") MultipartFile[] files
){}
```

而当操作变为编辑时，框架内部会自动生成两个参数提交后台，分别为 `deleteDynamicTableFile` 和 `dynamicTableNewFileIndex` 。顾名思义， `deleteDynamicTableFile` 是动态表格在做编辑操作时所删除的后端文件地址路径，删除多个文件是用逗号 " , " 分隔，后端可以根据这个字段的值来对已存储的文件做删除。而 `dynamicTableNewFileIndex` 保存的是做编辑操作时新上传文件的顺序，后端可以根据这个顺序来对应提交上来的表格数据。

```
public String test(
  @RequestParam(required = false, name = "tableField") String tableField,
```

```
  @RequestParam(required = false, name = "files") MultipartFile[] files,
  @RequestParam(required = false, name = "deleteDynamicTableFile") String deleteDynamicTableFile,
  @RequestParam(required = false, name = "dynamicTableNewFileIndex") String dynamicTableNewFileIndex
){}
```

源码分析：

```
<!--动态表格-->
<template v-if="item.type === \'dynamicTable\'">
    <el-card class="box-card" >
        <div slot="header" class="clearfix" style="padding-bottom:20px">
            <el-button style="float: right; padding: 3px 0" @click.prevent="addRow(item)" type="text">新增记录</el-button>
        </div>
        <el-table :data="formData[item.field]" height="290" border stripe ref="dynamicTable" :style="tableStyle(item)">
            <el-table-column v-for="col in item.cols" :key="col.prop" :prop="col.prop" align="center"
                             :label="typeof col.label === \'object\' ? col.label.value : col.label" :width="col.width">
                <template slot-scope="scope">
                    <el-form-item :prop="item.field+'.' + scope.$index + '.'+col.prop"
                                  :rules="dynamicTableRules[col.prop]">
                        <!--输入-->
                        <el-input v-if="col.type === \'input\'" v-model="scope.row[col.prop]" class="table_input_inner"
                                  style="margin-top: 15px;" :controls=false size="mini" v-bind="col.props">
                        </el-input>
                        <!--文本框-->
                        <el-input v-if="col.type === \'textarea\'" v-model="scope.row[col.prop]"
                                  type="textarea" :rows="3"
                                  style="margin-top: 10px;" :controls=false size="mini" v-bind="col.props">
                        </el-input>
                        <!--下拉框-->
                        <el-select v-if="col.type === \'select\'"  class="table_input_inner"
                                   v-model="scope.row[col.prop]"   size="mini"
                                   style="width:100%;display:block;margin-top: 15px;" v-bind="col.props">
                            <el-option v-for="option in col.options" :label="option.label" :value="option.value"></el-option>
                        </el-select>
                        <!--单选-->
                        <el-switch v-if="col.type === \'switch\'"  class="table_input_inner"
                                   style="margin-top: 15px;"
                                   v-model="scope.row[col.prop]" v-bind="col.props" size="mini"
                                   >
                        </el-switch>
                        <!--日期-->
                        <el-date-picker v-if="col.type === \'date\'" align="center"  class="table_input_inner"
                                        size="mini"
                                        style="margin-top: 15px;"
                                        v-model.trim="scope.row[col.prop]"
                                        v-bind="col.props"></el-date-picker>
                        <!--时间-->
                        <el-time-picker v-if="col.type === \'time\'" align="center"  class="table_input_inner"
                                        size="mini"
                                        style="margin-top: 15px;"
                                        v-model.trim="scope.row[col.prop]"
                                        v-bind="col.props"></el-time-picker>
                    </el-form-item>
                    <!--文件-->
                    <el-upload v-if="col.type === \'upload\'" class="upload-demo"
                               :ref="'upload_' + scope.$index"
                               action=""
                               multiple
```

```
                        :limit="1"
                        :name = col.prop
                        :file-list="dynamicTableFileList[scope.$index]"
                        :http-request="handleTableAddUpload"
                        :on-change = "function(file,fileList){return handleDynamicTableChange(file,fileList,scope.$index,item.
field,col.prop)}"
                        :on-preview="handlePictureCardPreview"
                        :on-remove="function(file,fileList){return handleDynamicTableRemove(file,fileList,scope.$index,col.
prop)}"
                        v-bind="col.props">
                    <el-button size="small" type="primary">点击上传</el-button>
                </el-upload>
            </template>
        </el-table-column>
        <el-table-column  label="操作" width="80" align="center">
            <template slot-scope="scope">
                <el-button @click.native.prevent="deleteRow(scope.$index, formData[item.field],item.field)"
                        type="text" size="small">
                    删除
                </el-button>
            </template>
        </el-table-column>
      </el-table>
    </el-card>
  </template>
```

动态表格嵌在原先的 `formDialog.js` 的 `item` 中，先判断外部传入的type是否为 `dynamicTable` ，是的话表格就开始生成。表格的数据还是保存在formData里面，方便提交。表格的列数据通过cols循环生成，单列的col所设置的表头参数会一一传递到这里。

表头设置完成之后，就是要生成该列所具体要实现的功能组件，有输入框，下拉框等等，也都是通过外部的设置的type来进行判断。不过这里会产生一个问题，在表格内外部的表单验证会失效，为了解决这个问题，我们需要在列上再套一层 `el-form-item` ,传入相应的rules绑定到对应的组件上即可实现表格内的数据验证。

数据验证通过之后，就开始处理提交时的数据，这里分无文件上传与有文件上传两种情况。无文件上传可以直接保存在原先框架的formData中，用 `field` 作为key。但是此时表格的数据是一串数组，所以在提交前需要将数组JSON化：

```
//没有文件情况下 需要把数据json化提交
for (var f in _formData) {
  var type = typeof _formData[f]
  if (type == "object" && Array.isArray(_formData[f]) && _formData[f].length > 0) {
    _formData[f] = JSON.stringify(_formData[f]);
  }
}
```

当有文件上传的时候，文件上传upload的数据是独立于formData之外的，这就造成了表格的行数据在做编辑操作时候难以跟文件数据对应。所以需要为其定义一种数据格式，把文件跟表格里面的数据联系起来：

```
//本地照片上传
this.formData[field][index][propName] = file.uid

//存入文件模型中
let fileModel = {
        "type": "local",
        "filePath": "",
        "uuid": file.uid,
        "index": index,
        "prop": propName,
        "fileName": file.name,
```

```
        ”fileObject”: file.raw
}

this.dynamicTableFilesModel[index] = fileModel;
```

本地文件新上传的时候，将文件的uid存入到对应index的表格行数据中，同时设计的文件模型数据中的uuid也存有该数据，这样的话，在提交时候就可以根据这个唯一标识来确定这个文件是对应表格中的第几行数据。

除此之外文件提交比较特殊的是需要用到formdata的set方法，对于表格里面的数组数据同样需要进行JSON化的处理。

右下方多按钮功能

要点一：在ref同级下新增 `buttons` 具体配置如下 `type` 不填的话默认是 `primary` 就是蓝色，`url` 不填的话就是默认 `submitUrl`

要点二： `buttons` 可以不写

要点三： `buttons` 写了就可以不写 `submitUrl` ，因为具体url都写在 `buttons` 中了

要点四：不论你新增几个按钮，都会默认带一个取消的白色按钮

```
dialogs:[{
                ref: ’editDialog’,
                title: ’编辑’,
                buttons:[{
                    label: ’测试1’,
                    type: ’primary’,
                    url: ’http://localhost:12006/WS/api/addForm1’,
                },{
                    label: ’测试2’,
                    type: ’success’,
                    url: ’http://localhost:12006/WS/api/addForm2’,
                },{
                    label: ’测试2’,
                    url: ’http://localhost:12006/WS/api/addForm3’,
                }],
                items: [
                    {
                        label: ’标题’,
                        type: ’input’,
                        field: ’title’,
                        readonly: true,
                        props: {
                            placeholder: ’请输入标题’
                        },
                        rules: [{required: true, message: ’请输入用户名’, trigger: ’blur’}],
                    },
                    {
                        label: ’创建时间’,
                        type: ’input’,
                        field: ’createTime’,
                        props: {
                            placeholder: ’请输入创建时间’
                        },
                    },
                    {
                        label: ’性别’,
```

```
                type: 'input',
                field: 'sex',
                props: {
                    placeholder: '请输入性别'
                },
                // rules: [{pattern: /^1[3456789]\d{9}$/, message: '请输入正确的手机', trigger: 'blur'}],
            }
        ],
        paramFields: ['opId'],
        enumMap: {
            sex: GLOBAL_ENUM.SEX
        },
        requireRow: true,
        submitUrl: "http://localhost:12006/WS/api/addForm",
        manual: true,
        handler: function (innerVm, formData, rawData, cb, ref, url) {
            console.log("test edit handler")
            console.log(formData);
            console.log(innerVm, formData, rawData, cb, ref, url);
            _this.$message.error("配置manual改为主动提交至后台，这里演示无法提交")
        }
    }],
```

实现思路：在formDialog.js中循环生成 `buttons` ，然后把 `url` 带给 `confirm` 方法,在 `confirm` 方法中有一个 `formData` 变量，利用 `formData['submitUrl'] = url` 的代码把 `url` 赋给 `formData` 这个json对象，然后formDialog.js会在 `confirm` 中emit base.js的 `onDialogValid` ，在该方法中会判断 `formData['submitUrl']` 是否存在,若存在则利用这个变量去替换当前的 `url` ，若不存在就会使用当前的 `url` ，其实就是你自己写的 `dialogs` 中的 `submitUrl` 也就是默认提交的对象当然如果写了buttons其实就不用去写submitUrl

源码分析

html代码中的rules是在formDialog.js中的created中处理的，template并没有直接处理

base.js的confirm是提交，但是真正的ajax请求在base.js的onDialogValid中