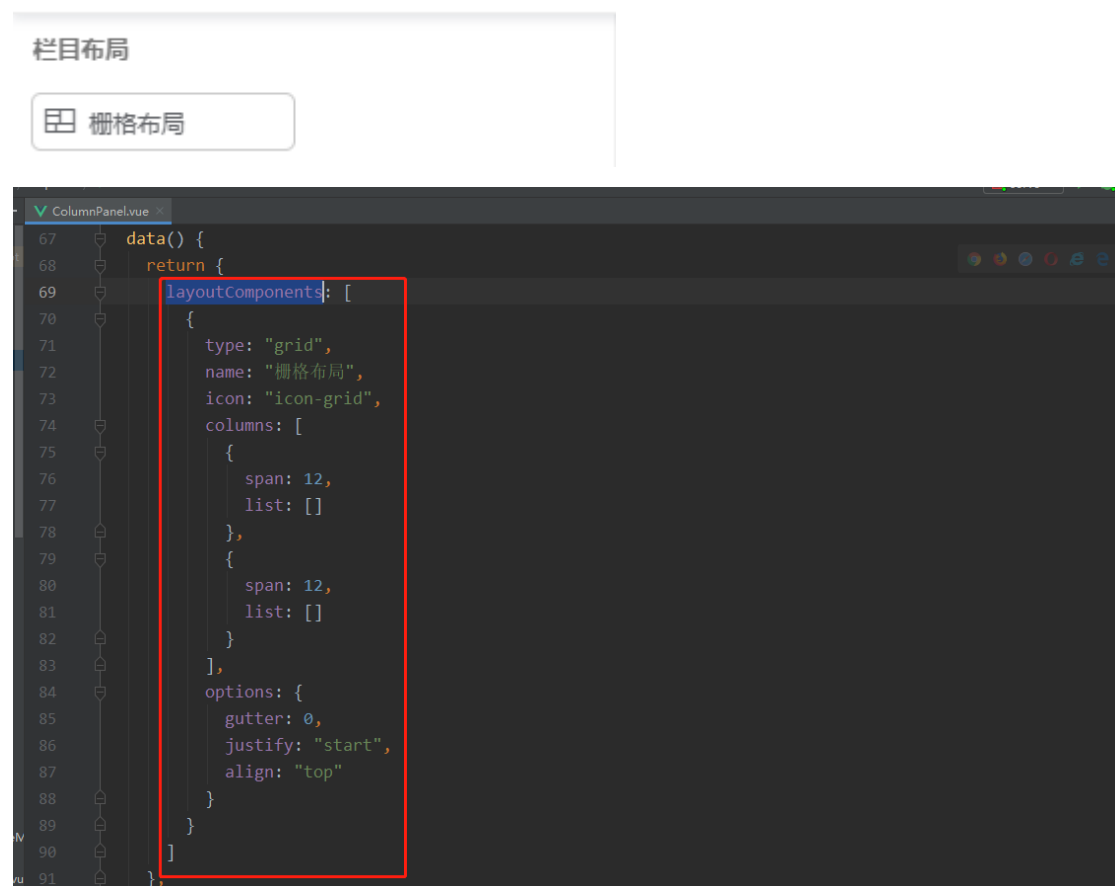


组件设计开发流程

1. 组件定制（自行开发需会 Vue 开发）

1.1 门户设计栏目布局，以栅格布局为例

ColumnPanel.vue 进行布局组件添加的 vue



layoutComponents: 布局组件数组，每个大括号中的数据都对应一个布局组件。

必须存在的属性：

type: 组件的类型，即英文名称

name: 组件名称，栏目布局中将显示该名称

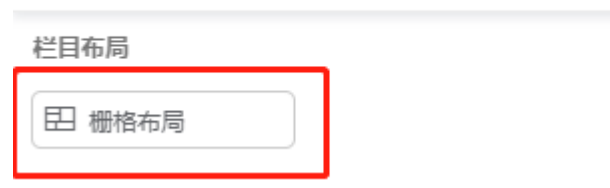
icon: 组件图标，栏目布局中将显示该图标

options: 组件参数，可传空对象

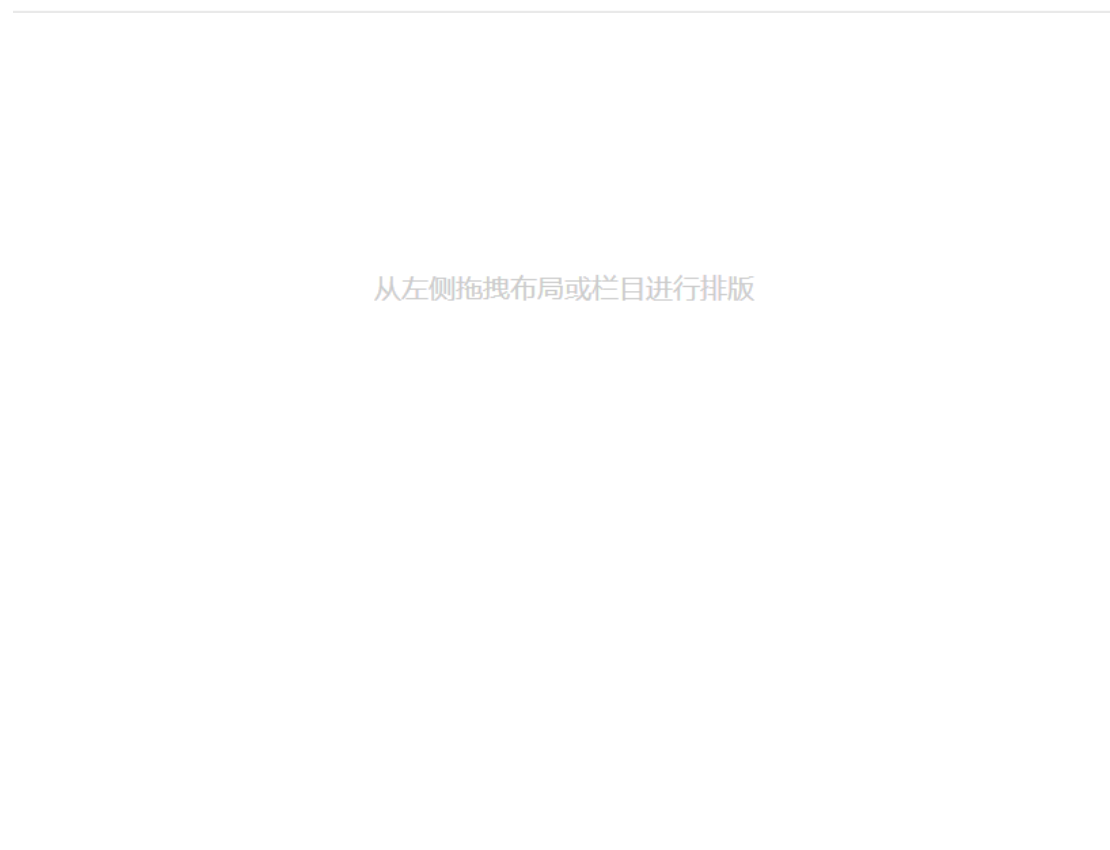
其它参数: 根据实际情况的需要进行添加修改

参数添加完成后将会在“栏目布局”中新增出一个布局组件

← 返回 | 门户首页设计器



DesignPanel.vue 中部组件排版 vue



根据选中的组件类型进行判断执行显示的 vue

```
13 @add="handleWidgetAdd"
14 >
15 <transition-group name="fade" tag="div" class="widget-form-list">
16   <template v-for="(element, index) in data.list.filter(d => d.key)">
17     <template v-if="element.type == 'grid'">
18       <design-grid-layout
19         :key="element.key"
20         :data="data"
21         :select.sync="selectWidget"
22         :index="index"
23         :element="element"
24       />
25     </template>
26     <template v-else>
27       <design-form-item
28         :key="element.key"
29         :element="element"
30         :select.sync="selectWidget"
31       />
32     </template>
33   </template>
34 </transition-group>
35
36 </script>
37
38 import Draggable from "vuedraggable";
39 import DesignFormItem from "@/components/portal/DesignFormItem.vue";
40 import DesignGridLayout from "@/components/portal/DesignGridLayout.vue";
```

接下来添加与组件类型相对应的 vue 文件

```
1 <template>
2   <div class="widget-form-item" @click.capture="handleSelectWidget(index)">
3     <div class="drag-widget" title="拖拽" v-if="selectWidget.key == element.key">
4       <i class="icon-drag"></i>
5     </div>
6     <div class="widget-view-action" v-if="selectWidget.key == element.key">
7       <i class="icon-trash" title="删除" @click.stop="handleWidgetDelete(index)"></i>
8     </div>
9     <el-row
10       class="widget-col"
11       v-if="element && element.key"
12       type="flex"
13       :class="{active: selectWidget.key == element.key}"
14       :gutter="element.options.gutter ? element.options.gutter : 0"
15       :justify="element.options.justify"
16       :align="element.options.align"
17     >
18       <el-col
19         v-for="(col, colIndex) in element.columns"
20         :key="colIndex"
21         :span="col.span ? col.span : 0"
22       >
23         <draggable
24           v-model="col.list"
25           :no-transition-on-drag="true">
```

LayoutConfigPanel.vue 中添加对应的栅格布局配置信息

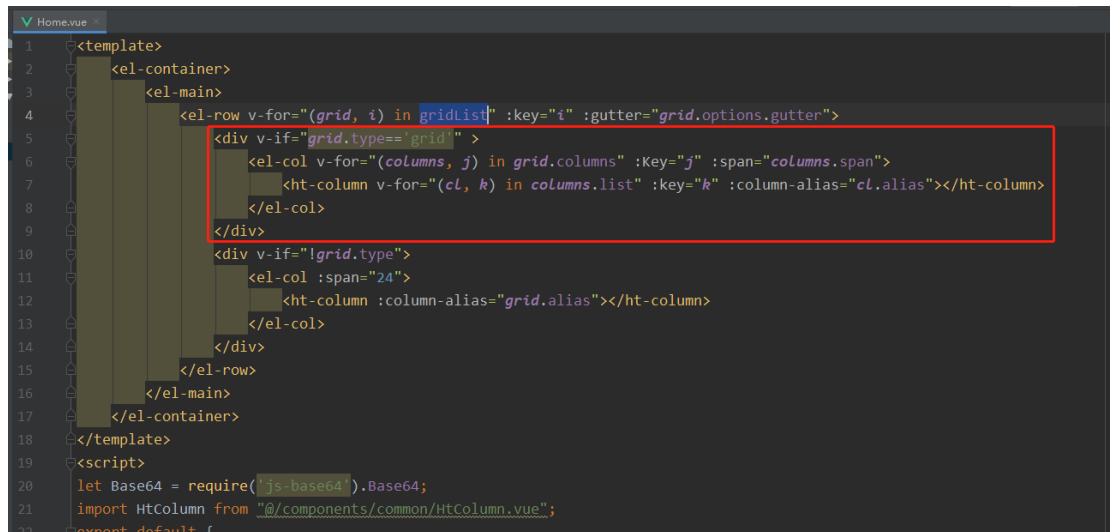
```

29 <el-container>
30   <el-header>栅格设计</el-header>
31   <el-main class="config-content">
32     <div v-if="!data || !data.type" class="field-empty">选择一个栅格布局进行设置</div>
33     <el-form label-position="top" v-if="data">
34       <template v-if="data.type == 'grid'">
35         <el-form-item label="栅格间隔: ">
36           <el-input type="number" v-model.number="data.options.gutter"></el-input>
37         </el-form-item>
38         <el-form-item label="列配置项: ">
39           <draggable
40             tag="ul"
41             :list="data.columns"
42             v-bind="{group:{ name:'options'}, ghostClass: 'ghost',handle: '.drag-item'}"
43             handle=".drag-item"
44           >
45             <li v-for="(item, index) in data.columns" :key="index">
46               <i class="drag-item" style="...">
47                 <i class="iconfont icon-draggable"></i>
48               </i>
49               <el-input
50                 placeholder="1到24的整数"
51                 size="mini"
52                 style="..."
53                 type="number"

```

对应布局类型的前端代码中 home.vue 中新增对应的 vue 组件，如栅格布局使用的 Htcolumn.vue 组件。

注：三种布局类型都需要在 home.vue 中添加对应代码



```
1 <template>
2   <el-container>
3     <el-main>
4       <el-row v-for="(grid, i) in gridlist" :key="i" :gutter="grid.options.gutter">
5         <div v-if="grid.type=='grid'">
6           <el-col v-for="(columns, j) in grid.columns" :key="j" :span="columns.span">
7             <ht-column v-for="(cl, k) in columns.list" :key="k" :column-alias="cl.alias"></ht-column>
8           </el-col>
9         </div>
10        <div v-if="!grid.type">
11          <el-col :span="24">
12            <ht-column :column-alias="grid.alias"></ht-column>
13          </el-col>
14        </div>
15      </el-row>
16    </el-main>
17  </el-container>
18 </template>
19 <script>
20   let Base64 = require('js-base64').Base64;
21   import HtColumn from "@components/common/HtColumn.vue";
22   export default {
```

1.2 业务表单设计布局组件定制

布局字段示例：以 tab 布局为例

首先在 controlsConfig.js 文件中 layoutComponents 添加对应布局属性

必须存在的属性：

ctrlType: 组件的类型

name: 必须存在不用填值

desc: 组件的显示名称

icon: 组件的图标

isLayout: true 固定写法

其它参数根据实际需求进行新增修改

```
controlsConfigs.js
714 export const LayoutComponents = [
715   {
716     ctrlType: "tab",
717     name: "",
718     desc: "tab布局",
719     icon: "icon-tab",
720     isLayout: true,
721     columns: [
722       {
723         span: "标签页1",
724         list: []
725       }
726     ],
727     options: {
728       gutter: 0,
729       nextCheck: "",
730       type: "", //风格类型
731       justify: "start",
732       align: "top"
733     }
734   },
735   {
736     ctrlType: "grid",
737     name: "",
738     desc: "栅格布局",
```

添加完成后

← 返回 | 表单设计(adss)

布局字段

TAB tab布局

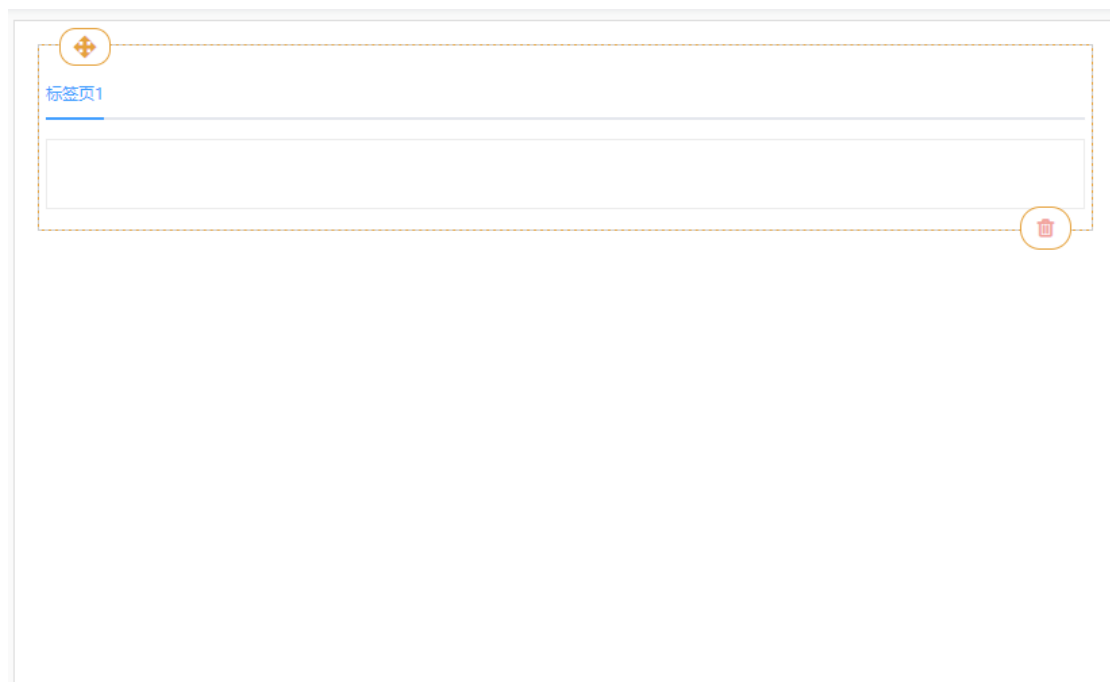
WidgetPanel.vue 中添加对应 vue 组件

```
WidgetPanel.vue
34 </template>
35 <template v-else-if="element.ctrlType == 'tab'">
36   <widget-table-layout
37     :key="element.key"
38     :data="data"
39     :select.sync="selectWidget"
40     :index="index"
41     :element="element"
42   />
43 </template>
44 <template v-else-if="element.ctrlType == 'page'">
45   <widget-pagination-layout
46     :key="element.key"
47     :data="data"
48     :select.sync="selectWidget"
49     :index="index"
50     :element="element"
51   />
52 </template>
53 <template v-else-if="element.ctrlType == 'pageSteps'">
54   <widget-page-steps-layout
55     :key="element.key"
56     :data="data"
57     :select.sync="selectWidget"
58     :index="index"
```

新增对应的 tab 组件代码

```
WidgetTableLayout.vue
1 <template>
2 <div class="widget-form-item" @click.capture="handleSelectWidget(index)">
3   <div
4     class="drag-widget"
5     title="拖拽"
6     v-if="selectWidget.key == element.key"
7   >
8     <i class="icon-drag"></i>
9   </div>
10  <div class="widget-view-action" v-if="selectWidget.key == element.key">
11    <i
12      class="icon-trash"
13      title="删除"
14      @click.stop="handleWidgetDelete(index)"
15    ></i>
16  </div>
17  <el-tabs
18    class="widget-col"
19    v-if="element && element.key"
20    :class="{ active: selectWidget.key == element.key }"
21    :v-model="activeName"
22    :justify="element.options.justify"
23    :tab-position="element.options.align"
24    :type="element.options.ctrlType"
25  >
```

WidgetPanel 中添加好组件后，布局或字段排版显示对应组件



WidgetTableLayout.vue 中添加对应的选项配置

```
91
92 <!--table选项卡配置-->
93 <template v-else-if="data && data.ctrltype == 'tab'">
94   <ht-form-item label-width label="风格类型">
95     <el-radio-group v-model="data.options.type">
96       <el-radio-button label="默认">默认</el-radio-button>
97       <el-radio-button label="card">选项卡</el-radio-button>
98       <el-radio-button label="border-card">卡片化</el-radio-button>
99     </el-radio-group>
100   </ht-form-item>
101   <ht-form-item label-width label="选项卡位置">
102     <el-radio-group v-model="data.options.align">
103       <el-radio-button label="top">顶部</el-radio-button>
104       <el-radio-button label="left">左侧</el-radio-button>
105       <el-radio-button label="right">右侧</el-radio-button>
106       <el-radio-button label="bottom">底部</el-radio-button>
107     </el-radio-group>
108   </ht-form-item>
109   <ht-form-item label-width>
110     <ht-checkbox
111       v-model="data.options.nextCheck"
112       :options="[ { key: 'true', value: '页签、切换前是否校验当前页' } ]"
113     />
114   </ht-form-item>
```

字段属性

表单属性

风格类型:

默认

选项卡

卡片化

选项卡位置:

顶部

左侧

右侧

底部

☐

页签、切换前是否校验当前页

标签配置项:

三

标签页1

—

添加标签

layoutTemplate.ftl 模板中添加对应的布局代码，最后初始化模板，新的布局组件就完成了


```

49 </el-row>
50 <#elseif layout.ctrlType=='tab'>
51 <el-tabs value="{{layout.key}}0" tab-position="{{layout.options.align}}" <#if layout.options.nextChec
52 <#list layout.columns as group>
53 <el-tab-pane label="{{group.span}}" name="{{layout.key}}{{group_index}}" ref="{{layout.key}}{{g
54 <template slot="label">
55 <#if group.span?length gt 10>
56 <el-tooltip class="item" effect="dark" content="{{group.span}}" placement="top-s
57 <a>{{group.span?substring(0,10)}}</a>
58 </el-tooltip>
59 </#if>
60 <#if group.span?length lt 10>
61 <span>{{group.span}}</span>
62 </#if>
63 </template>
64 <#list group.list as field>
65 <#if field.ctrlType=='grid'>
66 <el-row type="flex" :gutter="{{field.options.gutter}}" justify="{{field.options.
67 <#list field.columns as gridGroup>
68 <el-col :span="{{gridGroup.span}}" style="border: 1px solid #ccc;">
69 <#list gridGroup.list as gridField>
70 {{getFormItem(gridField,1)}}
71 </#list>
72 </el-col>

```

1.3 业务表单设计基础字段组件定制

基础字段示例：以多行文本为例

首先在 controlsConfig.js 中 basicComponents 添加对应的基础字段属性

必须存在的属性：

ctrlType: 组件的类型

name: 必须存在不用填值

desc: 组件的显示名称

icon: 组件的图标

其它属性根据实际需求添加

```
controlsConfig.js
1 export const basicComponents = [
2   {
3     ctrlType: "textarea",
4     name: "",
5     desc: "多行文本",
6     icon: "icon-textarea",
7     options: {
8       width: "100%",
9       isWidth: true,
10      defaultValue: "",
11      dataType: "string|text",
12      validateList: [],
13      validateType: "regex|min|max|required|row_unique|frontJSValidate|backendValidate",
14      advancedProperty: "isEditor|isInputEdit",
15      disabled: false,
16      placeholder: "",
17      basicsProperty: "placeholder",
18      formulasDiyJs: ""
19    }
20  },
21 ]
```

基础字段

 多行文本

WidgetFormItem.vue 中添加对应的 vue 组件

```
WidgetFormItem.vue
91 <template v-if="element.ctrlType == 'textarea'">
92   <ht-input
93     type="textarea"
94     :rows="element.options.height ? element.options.height : 2"
95     v-model="element.options.defaultValue"
96     :style="{
97       width: element.options.width ? element.options.width : '100%'
98     }"
99     :disabled="element.options.disabled"
100    :placeholder="
101      element.options.placeholder_zh || element.options.placeholder
102    "
103  />
104 </template>
```

HtInput.vue 代码

```
HtInput.vue
1 <template>
2   <div class="inputs" v-express>
3     <el-input
4       ref="elInput"
5       :size="size"
6       :style="inputWidth"
7       v-if="inputWriteable && type!='number'"
8       v-validate="inputValidate"
9       :name="inputName"
10      :placeholder="placeholder"
11      :clearable="clearable"
12      :show-password="showPassword"
13      :disabled="disabled"
14      :readonly="readonly"
15      v-model="inputVal"
16      :model-expression="modelExpression"
17      :type="type"
18      :rows="rowsVal"
19      :cols="colsVal"
20      :min="min"
21      :max="max"
22      :autosize="autosize"
23      :tabindex="tabindex"
24      :prefix-icon="prefixIcon"
25      :suffix-icon="suffixIcon"
```

多行文本:

BasicsProperty.vue 中可以添加已有的基本属性或者自行新增，基础属性的展示是由 controlsConfig.js 中 options 属性控制的

字段属性

表单属性

基础属性

▼

* ⓘ 绑定属性:

请选择

▼

字段标题:

多行文本

Q 国际化

控件类型:

多行文本

▼

标题宽度:

宽度: 100%

公式编辑:

编写公式

ⓘ 填写说明:

请输入内容

Q 国际化

AdvancedProperty.vue 中可以添加已有的高级属性或者自行新增，高级属性的展示是由 controlsConfig.js 中属性控制的

字段属性

表单属性

基础属性

高级属性

自定义控件宽度:

100%

标题字体样式:

加粗
 隐藏控件
 隐藏标题

提示信息:

编写内容

国际化

不可编辑

富文本:

字段校验(带*的支持EXCE

添加校验规则

L导入时校验:

fieldControl.ftl 中添加生成基础字段 vue 的代码

```

169 <#function getTextarea field type >
170     <#assign rtn>
171         <eip-textarea v-model="${getNgModel(field,type)}" model-name="${getNgModel(field,type)}"
172             placeholder="${field.options.placeholder}"
173             :permission="${getPermission(field,type)}"
174             attar="${getAtter(field,type)}"
175             isEditor="${field.options.isEditor}"
176             isInputEdit="${field.options.isInputEdit}"
177             initialFrameWidth="${field.options.initialFrameWidth}"
178             initialFrameHeight="${field.options.initialFrameHeight}"
179             style=""
180             type="${field.ctrlType}"
181             <#if field.options.textValue?if_exists>textValue='${field.options.textValue}' </#if>
182             <#if field.options.validate?if_exists && field.options.validate?length gt 0> :validate="${f
183             <#if field.options.formulasDiyJs?if_exists> v-formula="{value:${field.options.formulasDiyJ
184             <#if field.options.mapping?if_exists>v-mapping="data.${field.options.mapping}"</#if>
185         >
186             <span slot="labelDesc">${field.desc}</span>
187         </eip-textarea>
188     </#assign>
189     <#return rtn>
190 </#function>
  
```

```
fieldControl.kit
getTextarea
365 <!-- 注意不能加空格 -->
366 <#macro input field type >
367 <#switch field.ctrlType>
368 <#case 'textarea'><!-- 多行文本框-->${getTextarea(field,type)}
369 <#break>
```

在前端中新增对应基础字段的 vue 代码，如果属于新创建的 vue 文件
需在 OnlineForm.vue 中引入

```
EipTextarea.vue
1 <template>
2 <div class="inputs" ref="inputs">
3   <ht-input
4     :width="width"
5     style="width:100%;"
6     size="small"
7     :validate="inputValidate"
8     :name="inputName"
9     :placeholder="placeholder?placeholder:'请输入内容'"
10    v-model="inputVal"
11    :ref="this.atter"
12    type="textarea"
13    :disabled="!isEdit"
14    v-if="!isEditor && type!=='property-text'"
15    :permission="permission"
16  ></ht-input>
17  <span v-if="type=='property-text' && !isEditor">{{inputVal}}</span>
18  <vue-ueditor-wrap v-model="inputVal" :config="config" v-if="isEditor" ></vue-ueditor-wrap>
19  <!-- 用来控制富文本校验 -->
20  <ht-input
21    size="small"
22    :validate="inputValidate"
23    :permission="permission"
24    v-if="isEditor"
25    :name="inputName"
```

1.4 业务表单设计高级字段定制

高级字段示例：以按钮字段为例

首先在 controlsConfig.js 文件中 advanceComponents 添加对应字段属性

必须存在的属性：

ctrlType:组件的类型

name:必须存在不用填值

desc:组件的显示名称

icon:组件的图标

其它属性根据实际需求添加修改

```
controlsConfig.js
532 export const advanceComponents = [
533   {
534     ctrlType: "button",
535     name: "",
536     desc: "按钮",
537     icon: "icon-button",
538     options: {
539       noBindModel: false,
540       defaultType: "String",
541       basicsProperty: "onetextBtn",
542       bindEventjson: { name: "选择", icon: "", isShowInput: true }
543     },
544     noTitle: false,
545     noPlaceholder: false,
546     noTooltip: false
547   },
548 ]
```

高级字段



根据需求在 BasicsProperty.vue 添加基础属性配置，基础或高级属性由 controlsConfig.js 中的属性进行控制，在 AdvancedProperty.vue 中添加修改高级属性配置

```
BasicsProperty.vue
1863 <template v-if="isBasicsProperty"(field.options.basicsProperty, 'onetextBtn')">
1864   <h3 style="...">按钮事件绑定设置</h3>
1865   <ht-form-item label-width="190px" class="custDialog-item">
1866     <template slot="label">
1867       <el-tooltip content="目前只添加系统中需使用到的事件，如果不满足需求，需自行扩展事件。">
1868         <i class="property-tip icon-question" />
1869         <el-checkbox v-model="field.options.isBindBtn">绑定按钮调用JS方法</el-checkbox>
1870       </el-tooltip>
1871     </template>
1872     <el-button
1873       v-if="field.options.isBindBtn"
1874       class="btn-padding"
1875       icon="el-icon-plus"
1876       @click="autoRunJSScript(true)"
1877     >设置自定义脚本</el-button>
1878   </ht-form-item>
1879   <ht-form-item label-width="190px">
1880     <template slot="label">
1881       <el-tooltip content="是否显示输入框，如果显示需要绑定属性，js方法返回的值将填写到输入框中">
1882         <i class="property-tip icon-question" />
1883         <el-checkbox
1884           v-model="field.options.bindEventjson.isShowInput"
1885           @change="clickNoBindModel"
1886         >是否显示输入框</el-checkbox>
```

字段属性

表单属性

基础属性

* ② 绑定属性:

请选择

字段标题:

按钮

Q 国际化

控件类型:

按钮

标题宽度:

宽度: 100%

按钮事件绑定设置

☐ 绑定按钮调用JS方法:

☒ 是否显示输入框:

按钮名称:

选择

按钮图标

Q 选择图标

字段属性

表单属性

基础属性

高级属性

标题字体样式:

☐ 加粗

☐ 隐藏控件

☐ 隐藏标题

▼

②提示信息:

编写内容

Q 国际化

fieldControl.ftl 中添加生成高级字段 vue 的代码


```
fieldControl.ftl
getEipButton

149 <#function getEipButton field type>
150   <#assign name = util.getJsonByPath(field.options.bindEventjson,'name')>
151   <#assign icon = util.getJsonByPath(field.options.bindEventjson,'icon')>
152   <#assign isShowInput = util.getJsonByPath(field.options.bindEventjson,'isShowInput')>
153   <#assign alias = util.getJsonByPath(field.options.bindEventjson,'alias')>
154   <#assign rtn>
155     <eip-button
156       :isShowInput="${isShowInput?default(true)}"
157       <#if field.options.bindEventjson.isShowInput>
158         v-model="${getNgModel(field,type)}" model-name="${getNgModel(field,type)}"
159         :permission="${getPermission(field,type)}"
160         attr="${getAttr(field,type)}"
161       </#if>
162       icon="${icon}"
163       btnName="${name}"
164       htCustomScript="${field.options.script}" >
165     </eip-button>
166   </#assign>
167   <#return rtn>
168 </#function>

fieldControl.ftl
getEipButton
2 results

425 <#case 'eip-cascader'><!--级联-->${getEipCascader(field,type)}
426 <#break>
427 <#case 'button'>${getEipButton(field,type)}
428 <#break>
```

在前端中新增对应高级字段的 vue 代码，如果属于新创建的 vue 文件
需在 OnlineForm.vue 中引入

```
EipButton.vue

1 <template>
2   <div class="inputs" ref="inputs">
3     <ht-input
4       size="small"
5       v-if="isShowInput"
6       placeholder="请输入内容"
7       type="text"
8       v-model="inputValue"
9       :permission="permission"
10    >
11     <template slot="append">
12       <el-button btn="btn" style="..." size="small" type="primary" :icon="this.icon" @click="
13     </template>
14    </ht-input>
15    <el-button v-if="!isShowInput" btn="btn" style="..." size="small" type="primary" :icon="this.icon"
16  </div>
17 </template>
18 <script>
19   import ...
20   export default {
21     name: "eip-button",
22     props: [ "value", "name", "permission", "icon", "btnName", "attr", "htCustomScript", "isShowInput" ],
23     data() {
24       return {
25         unwatchAry: [],
26       }
27     }
28   }
29 </script>
```