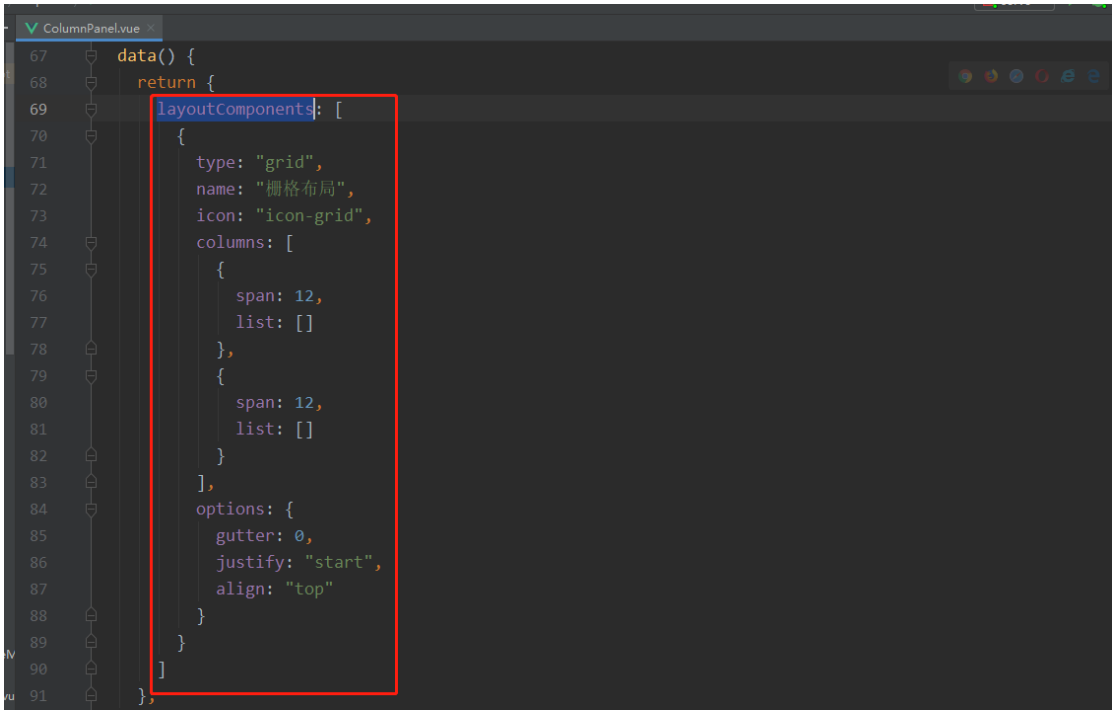
组件设计开发流程

1. 组件定制（自行开发需会 Vue 开发）

1.1门户设计栏目布局，以栅格布局为示例

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

DesignPanel.vue 中部组件排版 vue

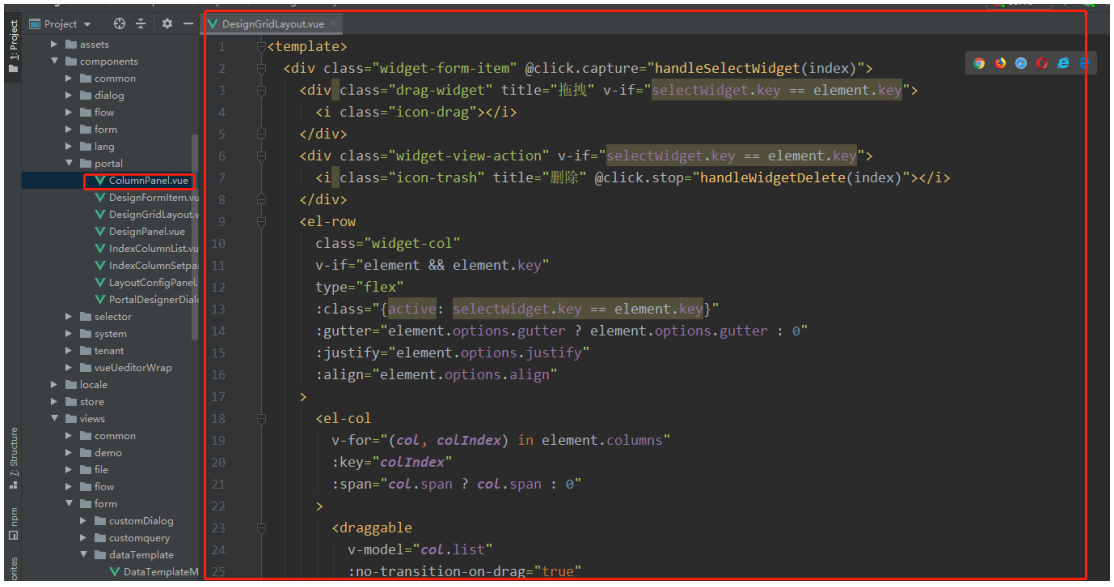
```
从左侧拖拽布局或栏目进行排版
```

根据选中的组件类型进行判断执行显示的 vue

```vue
DesignPanel.vue
      @add="handleWidgetAdd"
    >
      <transition-group name="fade" tag="div" class="widget-form-list">
        <template v-for="(element, index) in data.list.filter(d => d.key)">
          <template v-if="element.type == 'grid'">
            <design-grid-layout
              :key="element.key"
              :data="data"
              :select.sync="selectWidget"
              :index="index"
              :element="element"
            />
          </template>
          <template v-else>
            <design-form-item
              :key="element.key"
              :element="element"
              :select.sync="selectWidget"
```

```vue
<script>
import Draggable from "vuedraggable";
import DesignFormItem from "@/components/portal/DesignFormItem.vue";
import DesignGridLayout from "@/components/portal/DesignGridLayout.vue";
```

接下来添加与组件类型相对应的 vue 文件



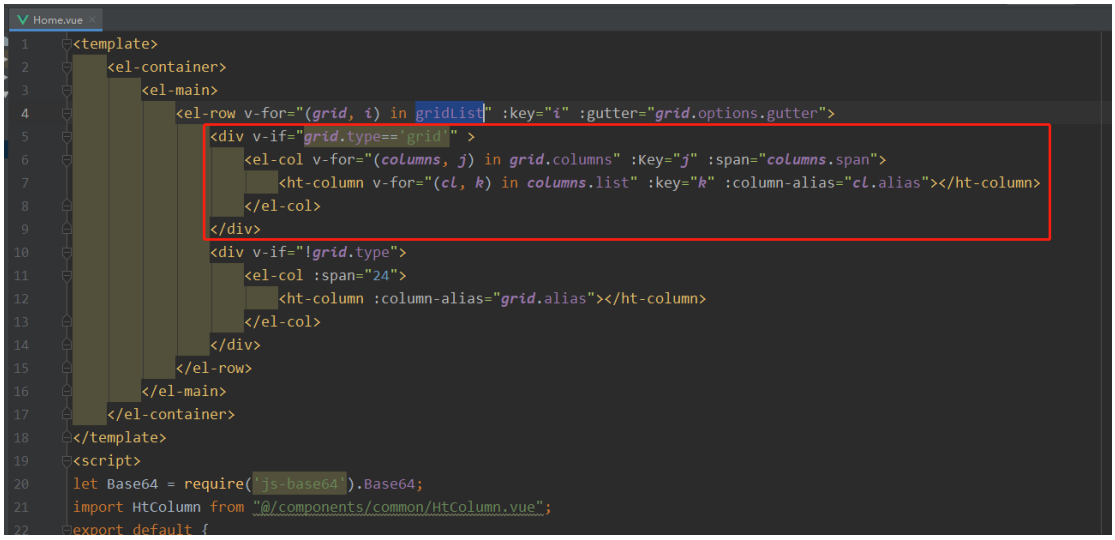
LayoutConfigPanel.vue 中添加对应的栅格布局配置信息

LayoutConfigPanel.vue

```
<el-container>
  <el-header>栅格设计</el-header>
  <el-main class="config-content">
    <div v-if="!data || !data.type" class="field-empty">选择一个栅格布局进行设置</div>
    <el-form label-position="top" v-if="data">
      <template v-if="data.type == 'grid'">
        <el-form-item label="栅格间隔：">
          <el-input type="number" v-model.number="data.options.gutter"></el-input>
        </el-form-item>
        <el-form-item label="列配置项：">
          <draggable
            tag="ul"
            :list="data.columns"
            v-bind="{group:{ name:'options'}, ghostClass: 'ghost',handle: '.drag-item'}"
            handle=".drag-item"
          >
            <li v-for="(item, index) in data.columns" :key="index">
              <i class="drag-item" style="...">
                <i class="iconfont icon-draggable"></i>
              </i>
              <el-input
                placeholder="1到24的整数"
                size="mini"
                style="..."
                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



1.2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

```
controlsConfig.js
export const layoutComponents = [
  {
    ctrlType: "tab",
    name: "",
    desc: "tab布局",
    icon: "icon-tab",
    isLayout: true,
    columns: [
      {
        span: "标签页1",
        list: []
      }
    ],
    options: {
      gutter: 0,
      nextCheck: "",
      type: "", //风格类型
      justify: "start",
      align: "top"
    }
  },
  {
    ctrlType: "grid",
    name: "",
    desc: "栅格布局",
```

添加完成后

WidgetPanel.vue 中添加对应 vue 组件

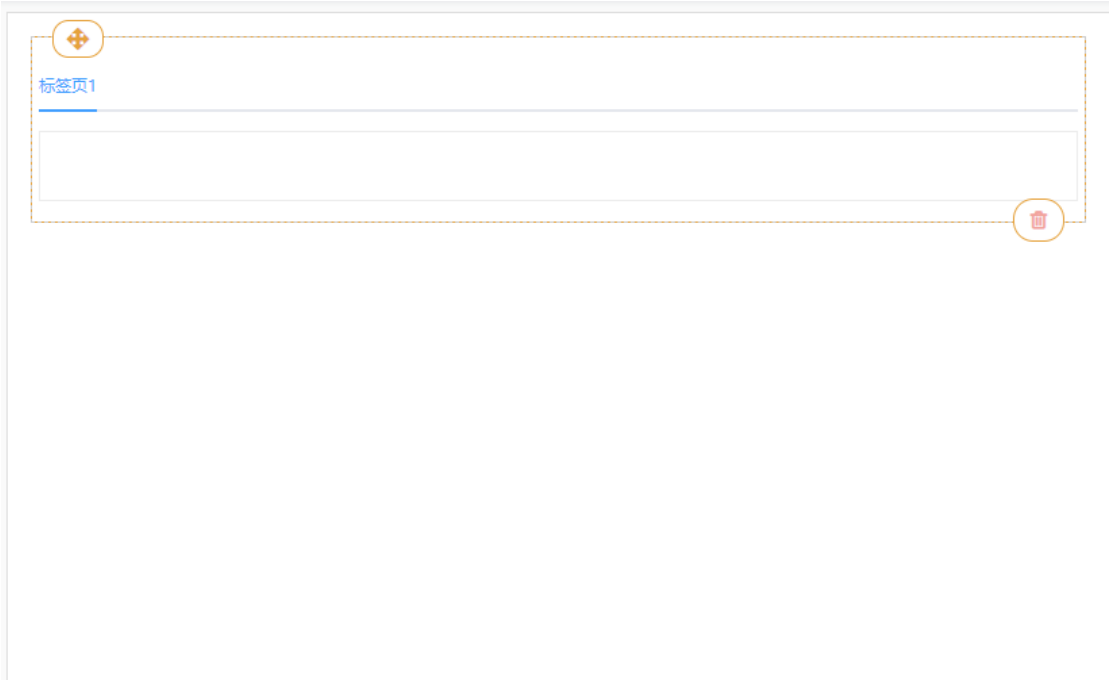
```
WidgetPanel.vue
    </template>
    <template v-else-if="element.ctrlType == 'tab'">
      <widget-table-layout
        :key="element.key"
        :data="data"
        :select.sync="selectWidget"
        :index="index"
        :element="element"
      />
    </template>
    <template v-else-if="element.ctrlType == 'page'">
      <widget-pagination-layout
        :key="element.key"
        :data="data"
        :select.sync="selectWidget"
        :index="index"
        :element="element"
      />
    </template>
    <template v-else-if="element.ctrlType == 'pageSteps'">
      <widget-page-steps-layout
        :key="element.key"
        :data="data"
        :select.sync="selectWidget"
        :index="index"
```

新增对应的 tab 组件代码

WidgetTableLayout.vue

```vue
<template>
  <div class="widget-form-item" @click.capture="handleSelectWidget(index)">
    <div
      class="drag-widget"
      title="拖拽"
      v-if="selectWidget.key == element.key"
    >
      <i class="icon-drag"></i>
    </div>
    <div class="widget-view-action" v-if="selectWidget.key == element.key">
      <i
        class="icon-trash"
        title="删除"
        @click.stop="handleWidgetDelete(index)"
      ></i>
    </div>
    <el-tabs
      class="widget-col"
      v-if="element && element.key"
      :class="{ active: selectWidget.key == element.key }"
      :v-model="activeName"
      :justify="element.options.justify"
      :tab-position="element.options.align"
      :type="element.options.ctrlType"
    >
```

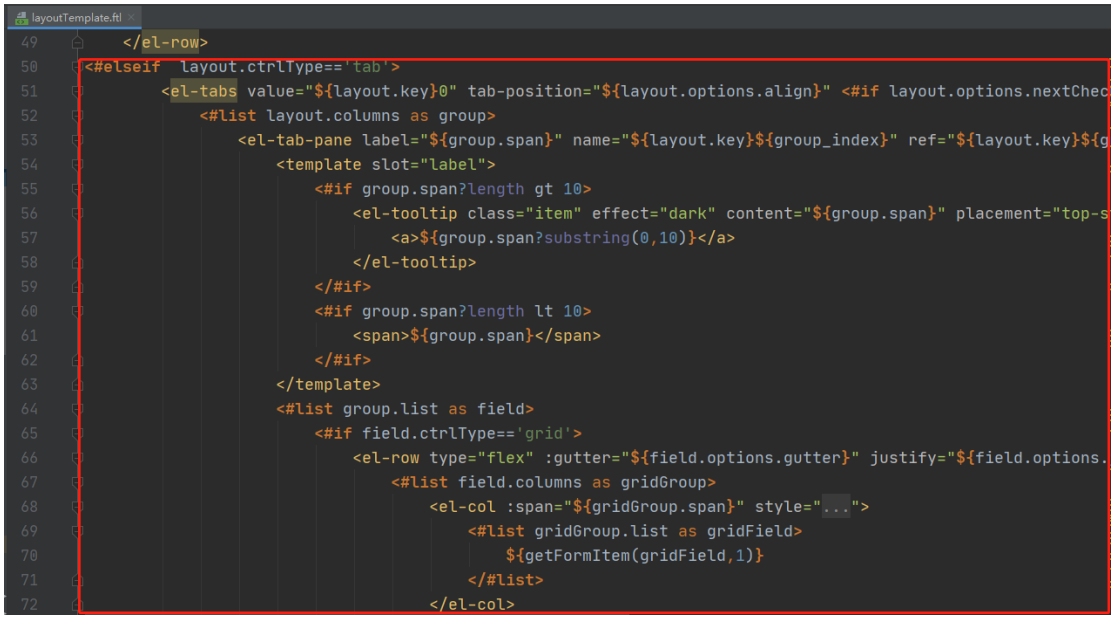
WidgetPanel 中添加好组件后，布局或字段排版显示对应组件



WidgetTableLayout.vue 中添加对应的选项配置

```vue
<!--table选项卡配置-->
<template v-else-if="data && data.ctrlType == 'tab'">
  <ht-form-item label-width label="风格类型">
    <el-radio-group v-model="data.options.type">
      <el-radio-button label>默认</el-radio-button>
      <el-radio-button label="card">选项卡</el-radio-button>
      <el-radio-button label="border-card">卡片化</el-radio-button>
    </el-radio-group>
  </ht-form-item>
  <ht-form-item label-width label="选项卡位置">
    <el-radio-group v-model="data.options.align">
      <el-radio-button label="top">顶部</el-radio-button>
      <el-radio-button label="left">左侧</el-radio-button>
      <el-radio-button label="right">右侧</el-radio-button>
      <el-radio-button label="bottom">底部</el-radio-button>
    </el-radio-group>
  </ht-form-item>
  <ht-form-item label-width>
    <ht-checkbox
      v-model="data.options.nextCheck"
      :options="[{ key: 'true', value: '页签、切换前是否校验当前页' }]"
    />
  </ht-form-item>
```

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



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

```
export const basicComponents = [
  {
    ctrlType: "textarea",
    name: "",
    desc: "多行文本",
    icon: "icon-textarea",
    options: {
      width: "100%",
      isWidth: true,
      defaultValue: "",
      dataType: "string|text",
      validateList: [],
      validateType: "regex|min|max|required|row_unique|frontJSValidate|backendValidate",
      advancedProperty: "isEditor|isInputEdit",
      disabled: false,
      placeholder: "",
      basicsProperty: "placeholder",
      formulasDiyJs: ""
    }
  },
```

WidgetFormItem.vue 中添加对应的 vue 组件

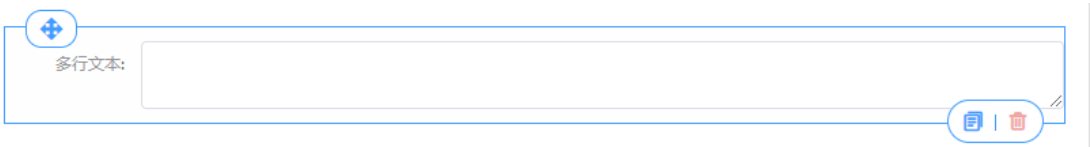
WidgetFormItem.vue

```
<template v-if="element.ctrlType == 'textarea'">
  <ht-input
    type="textarea"
    :rows="element.options.height ? element.options.height : 2"
    v-model="element.options.defaultValue"
    :style="{
      width: element.options.width ? element.options.width : '100%'
    }"
    :disabled="element.options.disabled"
    :placeholder="
      element.options.placeholder_zh || element.options.placeholder
    "
  />
</template>
```

HtInput.vue 代码

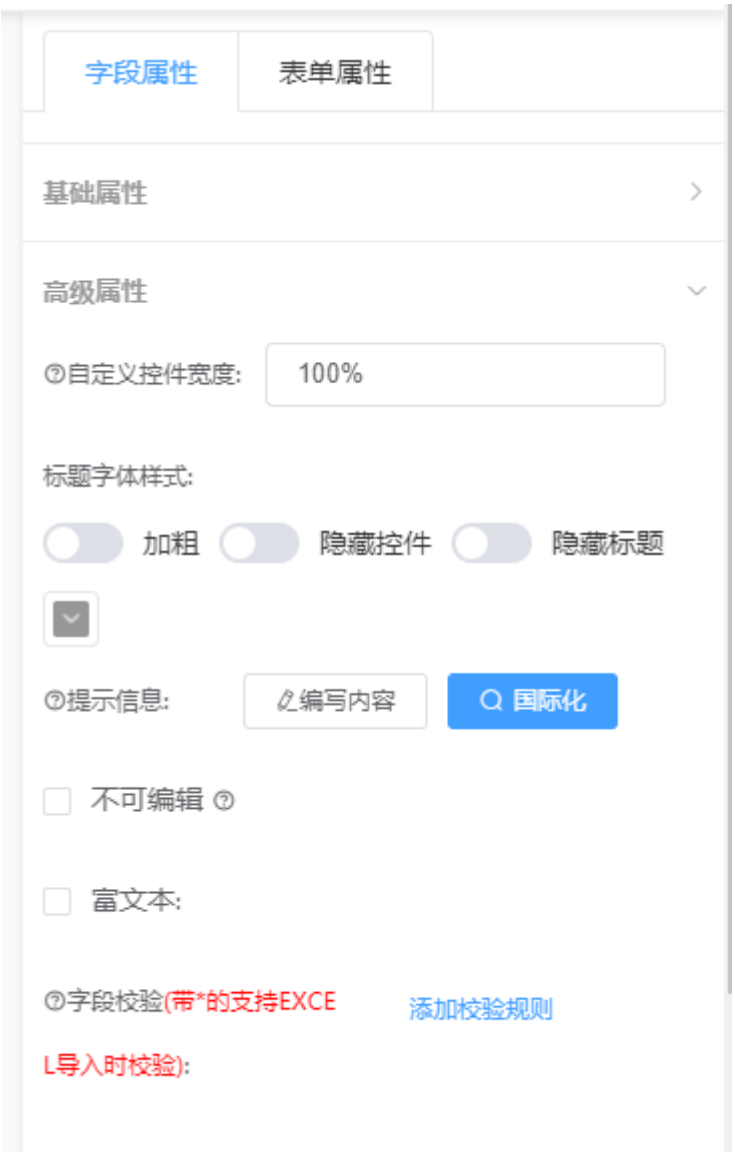
HtInput.vue

```vue
<template>
  <div class="inputs" v-express>
    <el-input
      ref="elInput"
      :size="size"
      :style="inputWidth"
      v-if="inputWriteable && type!='number'"
      v-validate="inputValidate"
      :name="inputName"
      :placeholder="placeholder"
      :clearable="clearable"
      :show-password="showPassword"
      :disabled="disabled"
      :readonly="readonly"
      v-model="inputVal"
      :model-expression="modelExpression"
      :type="type"
      :rows="rowsVal"
      :cols="colsVal"
      :min="min"
      :max="max"
      :autosize="autosize"
      :tabindex="tabindex"
      :prefix-icon="prefixIcon"
      :suffix-icon="suffixIcon"
```

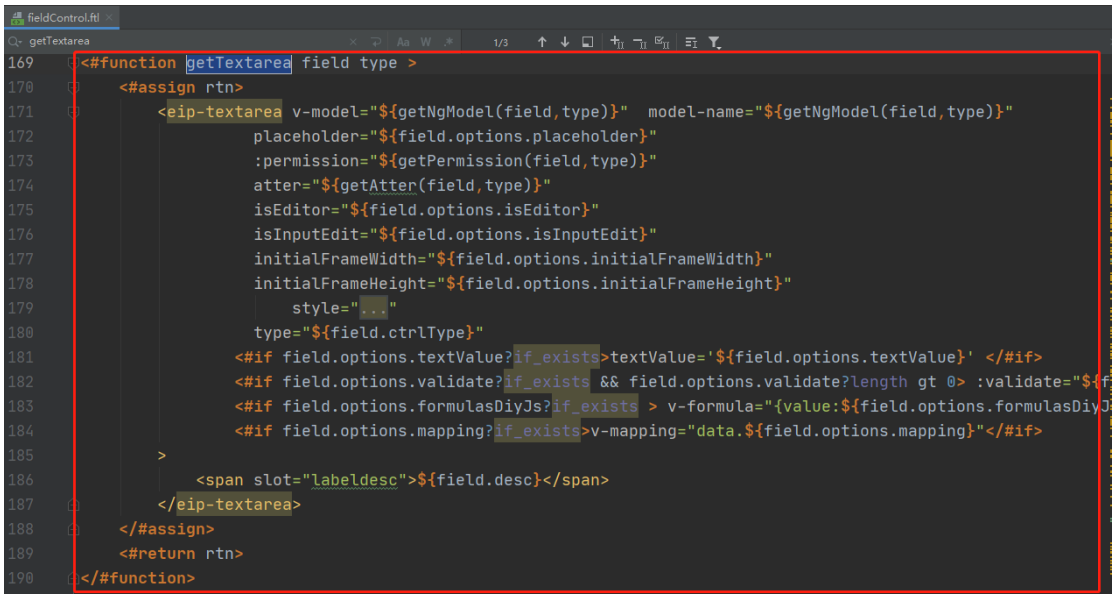


BasicsProperty.vue 中可以添加已有的基本属性或者自行新增，基础属性的展示是由 controlsConfig.js 中 options 属性控制的

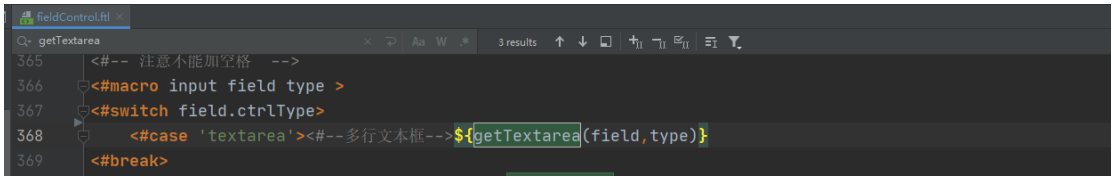
AdvancedProperty.vue 中可以添加已有的高级属性或者自行新增，高级属性的展示是由 controlsConfig.js 中属性控制的



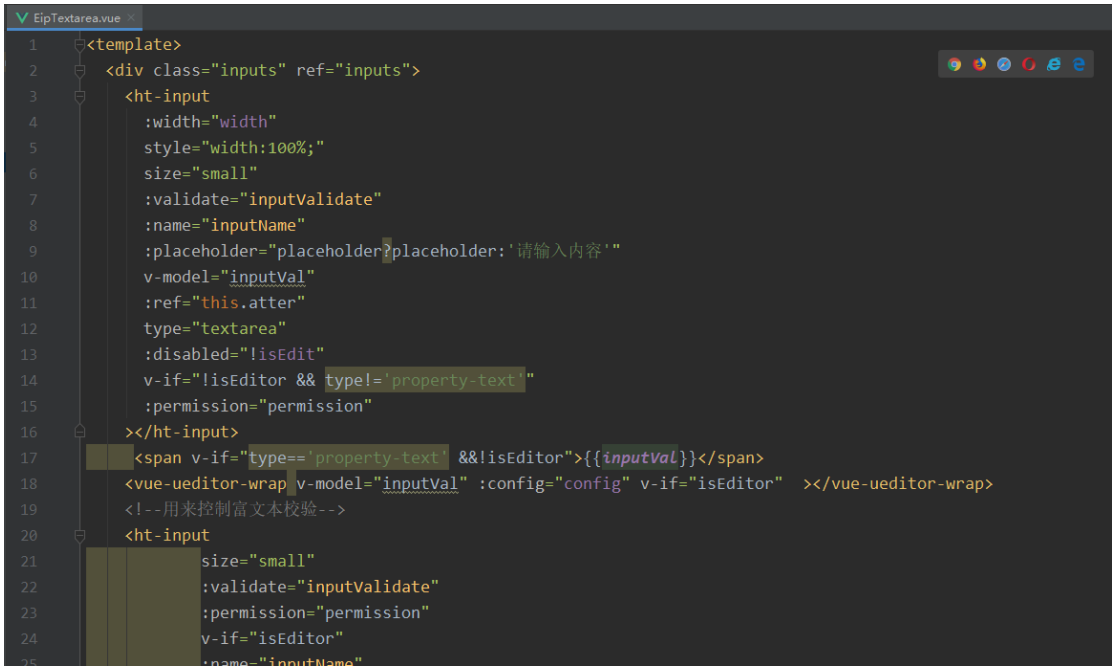
fieldControl.ftl 中添加生成基础字段 vue 的代码



```
<#function getTextarea field type >
    <#assign rtn>
        <eip-textarea v-model="${getNgModel(field,type)}"  model-name="${getNgModel(field,type)}"
                placeholder="${field.options.placeholder}"
                :permission="${getPermission(field,type)}"
                atter="${getAtter(field,type)}"
                isEditor="${field.options.isEditor}"
                isInputEdit="${field.options.isInputEdit}"
                initialFrameWidth="${field.options.initialFrameWidth}"
                initialFrameHeight="${field.options.initialFrameHeight}"
                    style="..."
                type="${field.ctrlType}"
               <#if field.options.textValue?if_exists>textValue='${field.options.textValue}' </#if>
               <#if field.options.validate?if_exists && field.options.validate?length gt 0> :validate="${f
               <#if field.options.formulasDiyJs?if_exists > v-formula="{value:${field.options.formulasDiyJ
               <#if field.options.mapping?if_exists>v-mapping="data.${field.options.mapping}"</#if>
        >
            <span slot="labeldesc">${field.desc}</span>
        </eip-textarea>
    </#assign>
    <#return rtn>
</#function>
```



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



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

```
export const advanceComponents = [
  {
    ctrlType: "button",
    name: "",
    desc: "按钮",
    icon: "icon-button",
    options: {
      noBindModel: false,
      defaultType: "String",
      basicsProperty: "onetextBtn",
      bindEventjson: { name: "选择", icon: "", isShowInput: true }
    },
    noTitle: false,
    noPlaceholder: false,
    noTooltip: false
  },
```

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

```
<template v-if="isBasicsProperty(field.options.basicsProperty, 'onetextBtn')">
  <H3 style="...">按钮事件绑定设置</H3>
  <ht-form-item label-width="190px" class="custDialog-item">
    <template slot="label">
      <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。">
        <i class="property-tip icon-question" />
        <el-checkbox v-model="field.options.isBindBtn">绑定按钮调用JS方法</el-checkbox>
      </el-tooltip>
    </template>
    <el-button
      v-if="field.options.isBindBtn"
      class="btn-padding"
      icon="el-icon-plus"
      @click="autoRunJSScript(true)"
    >设置自定义脚本</el-button>
  </ht-form-item>
  <ht-form-item label-width="190px">
    <template slot="label">
      <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">
        <i class="property-tip icon-question" />
        <el-checkbox
          v-model="field.options.bindEventjson.isShowInput"
          @change="clickNoBindModel"
        >是否显示输入框</el-checkbox>
```

fieldControl.ftl 中添加生成高级字段 vue 的代码

```
fieldControl.ftl
getEipButton                                                    1/2
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,'isShowIn
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="${getNgModel(fi
159                     :permission="${getPermission(field,type)}"
160                     atter="${getAtter(field,type)}"
161                     </#if>
162                     icon="${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 ...
24     export default {
25         name: "eip-button",
26         props: [ "value", "name", "permission","icon" ,"btnName" ,"attr","htCustomScript","isShowInput"],
27         data() {
28             return {
29                 unwatchAry: [],
```