芋道源码 2022-04-18

系统组件

1. 引入三方组件

除了 Element UI 组件以及项目内置的系统组件，有时还需要引入其它三方组件 。

1.1 如何安装

这里，以引入 vue-count-to 为例。在终端输入下面的命令完成安装：

```
## 加上 --save 参数，会自动添加依赖到 package.json 中去。
npm install vue-count-to --save
```

1.2 如何注册

Vue 注册组件有两种方式：全局注册、局部注册。

1.2.1 局部注册

在对应的 Vue 页面中，使用 `components` 属性来注册组件。代码如下：

```
<template>
  <countTo :startVal='startVal' :endVal='endVal' :duration='3000'></countTo>
</template>

<script>
import countTo from 'vue-count-to';
export default {
  components: { countTo }, // components 属性
  data () {
    return {
      startVal: 0,
      endVal: 2017
    }
  }
}
</script>
```

1.2.2 全局注册

① 在 main.js 中，全局注册组件。代码如下：

```
import countTo from 'vue-count-to'
Vue.component('countTo', countTo)
```

② 在对应的 Vue 页面中，直接使用组件，无需注册。代码如下：

```
<template>
  <countTo :startVal='startVal' :endVal='endVal' :duration='3000'></countTo>
</template>
```

2. 系统组件

项目使用到的相关组件。

2.1 基础框架组件

element-ui

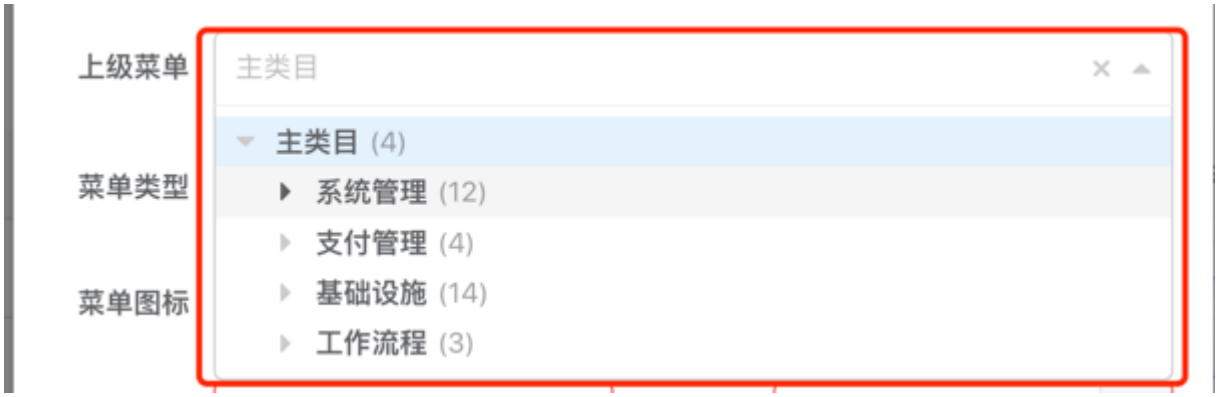
vue-element-admin

2.2 树形选择组件

vue-treeselect

在 menu/index.vue 的使用案例：

```
<el-form-item label="上级菜单">
  <treeselect v-model="form.parentId" :options="menuOptions" :normalizer="normal
              placeholder="选择上级菜单"/>
</el-form-item>
```

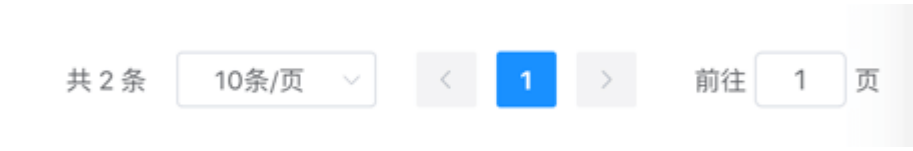


2.3 表格分页组件

`el-pagination` ，二次封装成 `pagination` 组件。

在 `notice/index.vue` 的使用案例：

```
<pagination v-show="total>0" :total="total" :page.sync="queryParams.pageNo" :lim
            @pagination="getList"/>
```



2.4 工具栏右侧组件

`right-toolbar`

在 `notice/index.vue` 的使用案例：

```
<right-toolbar :showSearch.sync="showSearch" @queryTable="getList"></right-toolb
```

2.5 文件上传组件

`file-upload`

2.6 图片上传组件

图片上传组件 `image-upload`

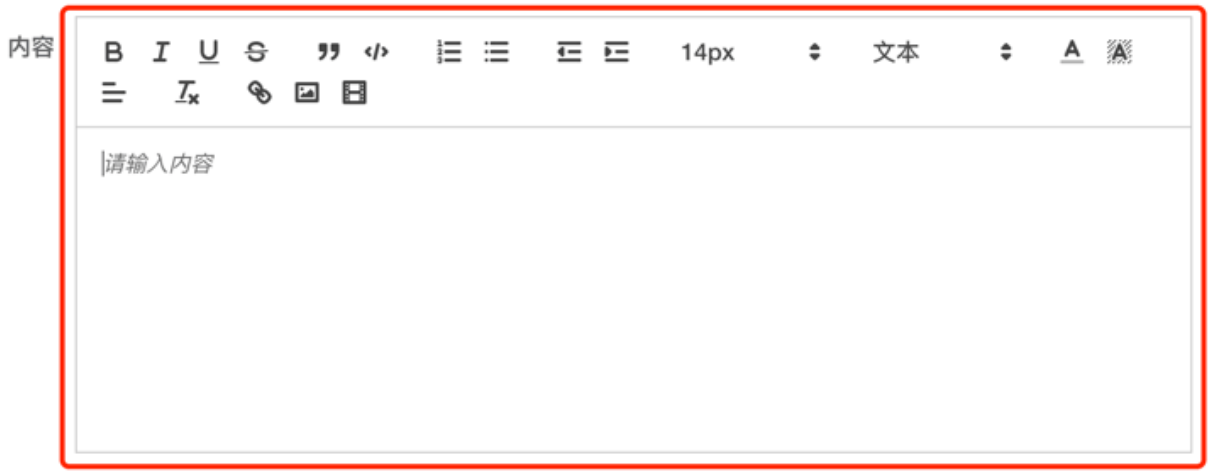
图片预览组件 image-preview

2.7 富文本编辑器

quill ，二次封装成 Editor 组件。

在 notice/index.vue 的使用案例：

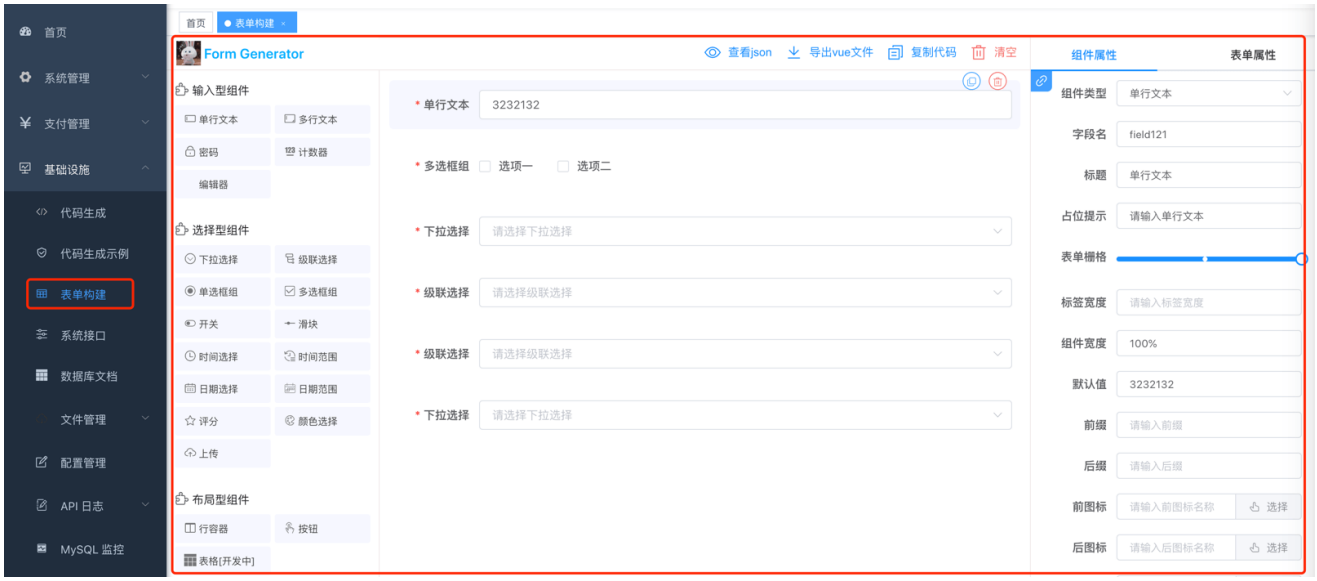
```
<el-form-item label="内容">
  <editor v-model="form.content" :min-height="192"/>
</el-form-item>
```



2.8 表单设计组件

① 表单设计组件 form-generator

在 build/index.vue 中使用，效果如下图：



② 表单展示组件 parser ，基于 form-generator 封装。

在 processInstance/create.vue 的使用案例：

```
<parser :key="new Date().getTime()" :form-conf="detailForm" @submit="submitForm"
```

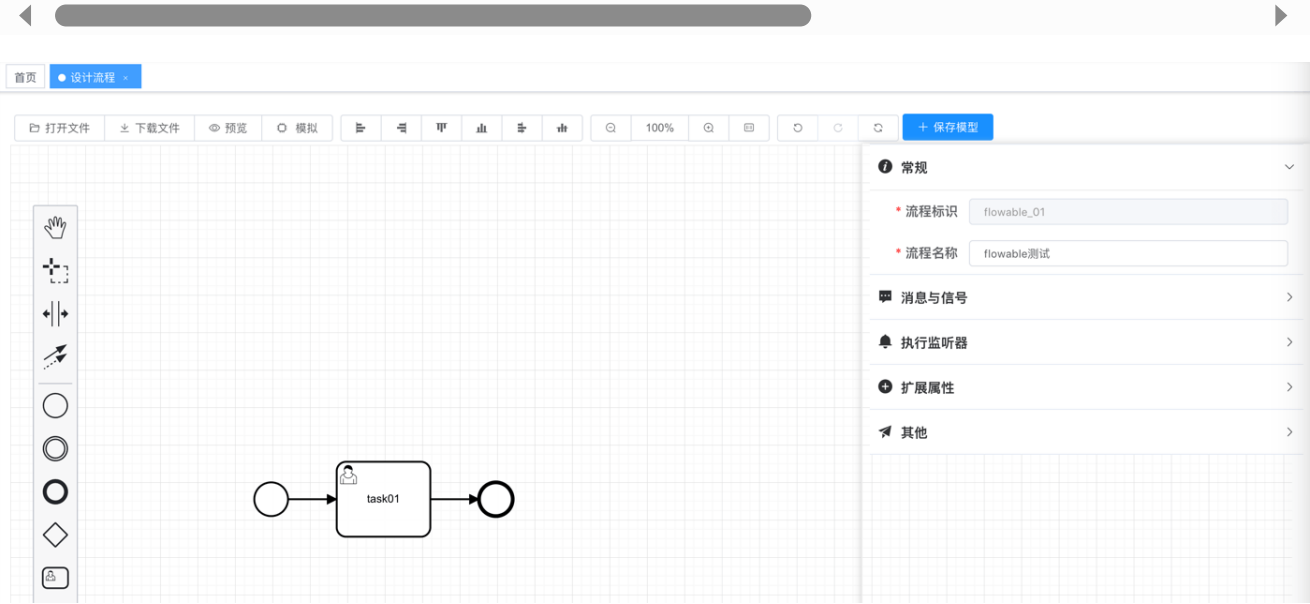
2.9 工作流组件

bpmn-process-designer ，二次封装成 bpmnProcessDesigner 工作流设计组件

① 工作流设计组件 my-process-designer ，在 bpm/model/modelEditor.vue 中使用案例：

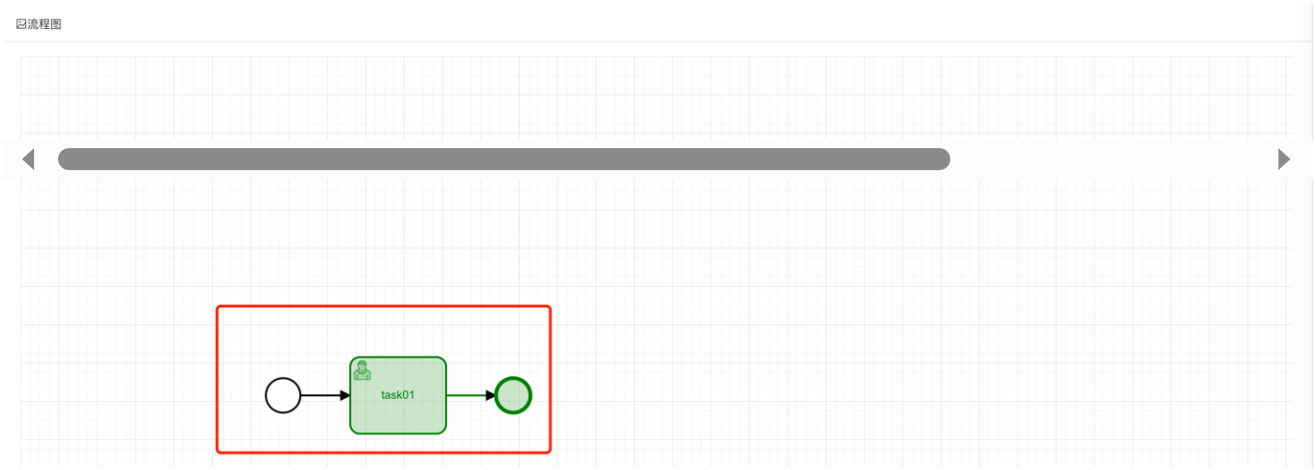
```
<!-- 流程设计器，负责绘制流程等 -->
<my-process-designer :key="`designer-${reloadIndex}`" v-model="xmlString" v-binc
  keyboard ref="processDesigner" @init-finished="initModeler"
  @save="save"/>

<!-- 流程属性器，负责编辑每个流程节点的属性 -->
<my-properties-panel :key="`penal-${reloadIndex}`" :bpmn-modeler="modeler" :pref
  :model="model" />
```



② 工作流展示组件 my-process-viewer ，在 bpm/model/modelEditor.vue 中使用案例：

```
<my-process-viewer key="designer" v-model="bpmnXML" v-bind="bpmnControlForm" :ac
          :processInstanceData="processInstance" :taskData="tasks" />
```



2.10 Cron 表达式组件

vue-crontab ，二次封装成 crontab 组件。

在 job/index.vue 的使用案例：

```
<crontab @hide="openCron=false" @fill="crontabFill" :expression="expression"></c
```

2.11 内容复制组件

clipboard ，使用可见 文档 。

在 codegen/index.vue 的使用案例：

```
<el-link :underline="false" icon="el-icon-document-copy" style="float:right"
         v-clipboard:copy="item.code"
         v-clipboard:success="clipboardSuccess">
    复制
</el-link>
```

3. 其它推荐组件

推荐一些其它组件，可自己引入后使用。

- Tree Table 树形表格：使用文档
- Excel 前端直接导出：使用文档
- CodeMirror 代码编辑器：使用文档
- wangEditor 文本编辑器：使用文档
- mavonEditor Markdown 编辑器：使用文档

4. 自定义组件

在 @/components 目录下，创建 .vue 文件，在通过 components 进行注册即可。

4.1 创建使用

新建一个简单的 a 组件来举例子。

① 在 @/components/ 目录下，创建 test 文件，再创建 a.vue 文件。代码如下：

```
<!-- 子组件 -->
<template>
  <div>这是a组件</div>
</template>
```

② 在其它 Vue 页面，导入并注册后使用。代码如下：

```
<!-- 父组件 -->
<template>
  <div style="text-align: center; font-size: 20px">
    测试页面
    <testa></testa> <!-- 3. 使用 -->
  </div>
</template>

<script>
import a from "@/components/a"; // 1. 引入
export default {
  components: { testa: a } // 2. 注册
};
</script>
```

4.2 组件通信

基于上述的 `a` 示例组件，讲解父子组件如何通信。

① 子组件通过 `props` 属性，来接收父组件传递的值。代码如下：

```
<!-- 子组件 -->
<template>
    <div>这是a组件 name:{{ name }}</div>
</template>

<script>
    export default {
        props: { // 1. props 的 name 进行接收
            name: {
                type: String,
                default: ""
            },
        }
    };
</script>
```

```
<!-- 父组件 -->
<template>
  <div style="text-align: center; font-size: 20px">
    测试页面
    <testa :name="name"></testa> <!-- 2. :name 传入 -->
  </div>
</template>

<script>
import a from "@/components/a";

export default {
  components: { testa: a },
  data() {
    return {
      name: "芋道"
    };
  },
};
</script>
```

② 子组件通过 `$emit` 方法，让父组件监听到自定义事件。代码如下：

```
<!-- 子组件 -->
<template>
  <div>
    这是a组件 name:{{ name }}
    <button @click="click">发送</button>
  </div>
</template>

<script>
export default {
  props: {
    name: {
      type: String,
      default: ""
    },
  },
  data() {
    return {
      message: "我是来自子组件的消息"
    };
  },
  methods: {
```

```
    click() {
      this.$emit("ok", this.message); // 1. $emit 方法，通知 ok 事件，message 是参
    },
  },
};
</script>

<!-- 父组件 -->
<template>
    <div style="text-align: center; font-size: 20px">
        测试页面
        <testa :name="name" @ok="ok"></testa>
        子组件传来的值 : {{ message }}
    </div>
</template>

<script>
import a from "@/components/a";

export default {
    components: { testa: a },
    data() {
        return {
            name: "芋道",
            message: ""
        };
    },
    methods: {
        ok(message) { // 2. 声明 ok 方法，监听 ok 自定义事件
            this.message = message;
        },
    },
};
</script>
```

← 字典数据 通用方法→