

# 1 Element UI 简介

基于 Vue 的一套桌面端组件库，提前封装好的 UI 模板，方便开发者快速搭建一个网站前端界面。

官网：<https://element.eleme.cn/>

## 2 Element UI 安装

首先创建 Vue 项目，给项目安装 Element UI 插件。

1、电脑上已经安装了 Vue 环境。

2、执行命令 vue ui

- 进入项目管理器



2、点击创建按钮



### 3、设置项目存放路径，点击回车



### 4、点击按钮

+ 在此创建新项目

### 5、输入工程名称，点击下一步



## 6、选择手动配置项目，点击下一步

选择一套预设

- ☐ 默认  
[Vue 2] babel, eslint
- ☐ Default preset (Vue 3)  
[Vue 3] babel, eslint
- ☒ 手动  
手动配置项目
- ☐ 远程预设  
从 git 仓库拉取预设

← 上一步      下一步 →

## 7、进行功能配置，点击下一步

Progressive Web App (PWA) Support  
Improve performances with features like Web manifest and Service workers [查看详情](#)

Router  
Structure the app with dynamic pages [查看详情](#)

Vuex  
Manage the app state with a centralized store [查看详情](#)

CSS Pre-processors  
Add support for CSS pre-processors like Sass, Less or Stylus [查看详情](#)

Linters / Formatters  
Check and enforce code quality with ESLint or Prettier [查看详情](#)

← 上一步      下一步 →

## 8、开启 history mode，点击创建项目

Use history mode for router? (Requires proper server setup for index fallback in production)  
By using the HTML5 History API, the URLs don't need the '#' character anymore. [查看详情](#)

← 上一步      ✓ 创建项目

## 9、创建项目

保存为新预设

×

预设名

 |

将功能和配置保存为一套新的预设

取消

创建项目，不保存预设

 保存预设并创建项目

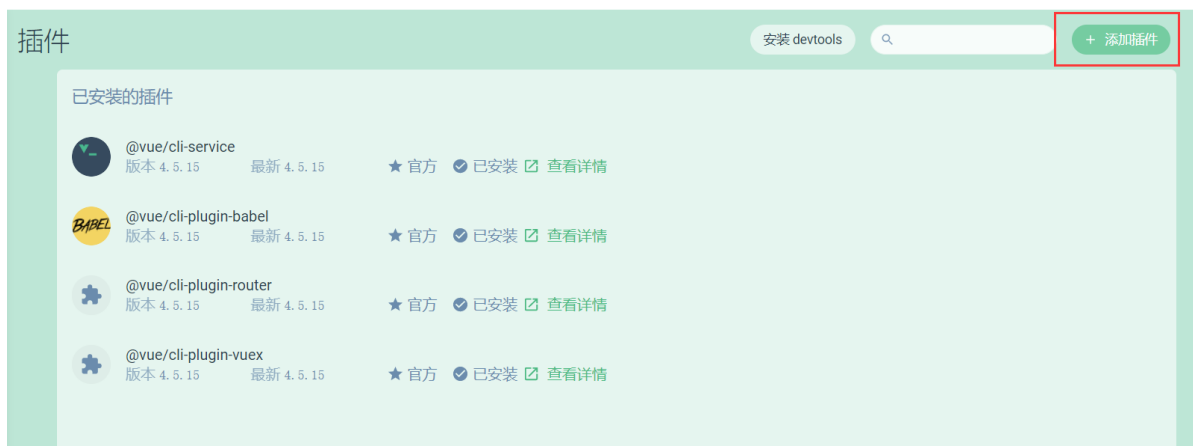
## 10、等待项目创建成功



## 11、点击插件



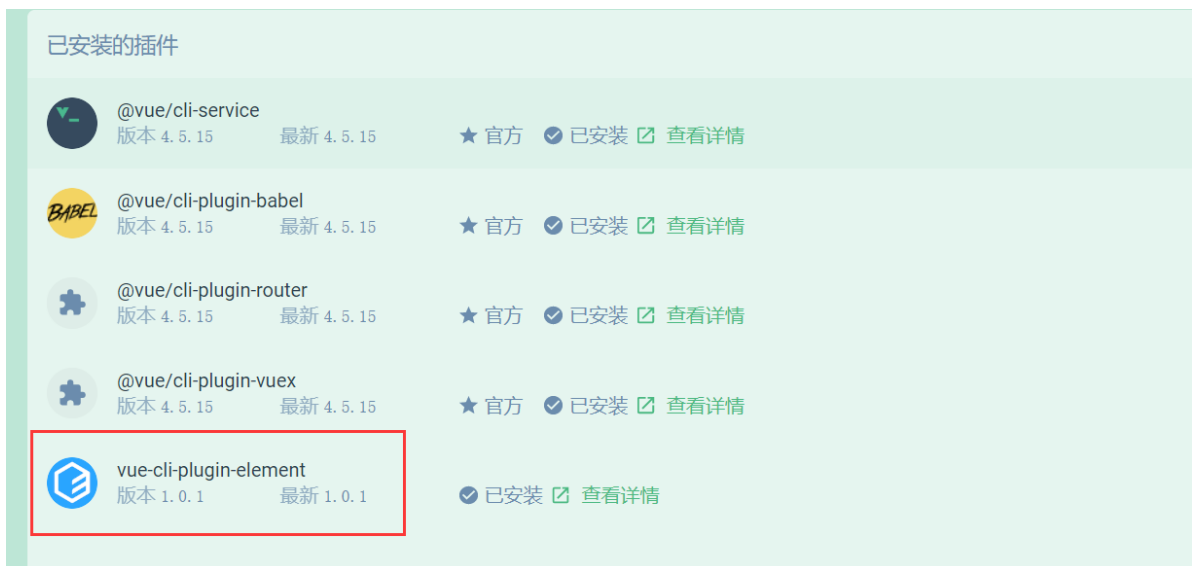
## 12、点击添加插件



## 13、搜索框输入 element，选中第一项，点击安装



## 14、安装成功，界面如下所示



## 15、启动项目，如果安装成功，会看到如下界面



If Element is successfully added to this project, you'll see an `<el-button>` below

el-button

## Welcome to Your Vue.js App

























For a guide and recipes on how to configure / customize this project,  
check out the [vue-cli documentation](#).

# 3 Icon 图标

提供了一套常用的图标集合，直接使用 `i` 标签结合 `class` 来使用：

```
<i class="el-icon-iconName"></i>
```

`el-icon-iconName` 为官网定义的图标名称，直接取官网查找对应的图标，修改 `class` 属性即可。

 el-icon-platform-eleme	 el-icon-eleme	 el-icon-delete-solid	 el-icon-delete	 el-icon-s-tools	 el-icon-setting
 el-icon-user-solid	 el-icon-user	 el-icon-phone	 el-icon-phone-outline	 el-icon-more	 el-icon-more-outline
 el-icon-star-on	 el-icon-star-off	 el-icon-s-goods	 el-icon-goods	 el-icon-warning	 el-icon-warning-outline
 el-icon-question	 el-icon-info	 el-icon-remove	 el-icon-circle-plus	 el-icon-success	 el-icon-error

## 4 Button 按钮

是 Element UI 提供的一组常用的操作按钮组件，直接使用封装好的 el-button ,比如：

```
<el-button>按钮</el-button>
```



基于 el-button 按钮，可以使用 type、plain、round、circle 属性对按钮进行修饰。

type 设置按钮的样式



```
<el-button>默认按钮</el-button>
<el-button type="primary">主要按钮</el-button>
<el-button type="success">成功按钮</el-button>
<el-button type="info">信息按钮</el-button>
<el-button type="warning">警告按钮</el-button>
<el-button type="danger">危险按钮</el-button>
```



plain 可以减弱按钮的背景颜色效果

```
<el-button plain>默认按钮</el-button>
<el-button type="primary" plain>主要按钮</el-button>
<el-button type="success" plain>成功按钮</el-button>
<el-button type="info" plain>信息按钮</el-button>
<el-button type="warning" plain>警告按钮</el-button>
<el-button type="danger" plain>危险按钮</el-button>
```



## round 用来给按钮设置圆角

```
<el-button round>圆角按钮</el-button>  
<el-button type="primary" round plain>主要按钮</el-button>  
<el-button type="success" round>成功按钮</el-button>  
<el-button type="info" round>信息按钮</el-button>  
<el-button type="warning" round>警告按钮</el-button>  
<el-button type="danger" round>危险按钮</el-button>
```

圆角按钮

主要按钮

成功按钮

信息按钮

警告按钮

危险按钮

## circle 将按钮设置为圆形

```
<el-button circle>默认按钮</el-button>  
<el-button type="primary" circle>主要按钮  
</el-button>  
<el-button type="primary" circle icon="el-  
icon-share"></el-button>  
<el-button type="success" circle>成功按钮  
</el-button>  
<el-button type="info" circle>信息按钮</el-  
button>  
<el-button type="warning" circle>警告按钮  
</el-button>  
<el-button type="danger" circle>危险按钮</el-  
button>
```



disabled 设置按钮的可用状态

```
<el-button type="primary" circle icon="el-  
icon-share"></el-button>  
<el-button type="primary" circle icon="el-  
icon-share" disabled></el-button>
```



loading 属性可用给按钮设置“加载中”的效果

```
<template>
```

```
<div>
  <el-button type="primary" circle
icon="el-icon-share" @click="test()"
:loading="loading"></el-button>
</div>
</template>

<script>
  export default {
    name: "Test",
    methods:{
      test(){
        this.loading = true;
        setTimeout(()=>{
          this.loading = false
        },3000)
      }
    },
    data(){
      return{
        loading:false
      }
    }
  }
</script>
```

size 属性可用设置按钮的大小, medium, small, mini

```
<el-button type="primary" size="medium">主要按钮</el-button>  
<el-button type="primary" size="small">主要按钮</el-button>  
<el-button type="primary" size="mini">主要按钮</el-button>
```

主要按钮

主要按钮

主要按钮

## 5 Link 超链接

文字超链接，使用 el-link 标签来实现

```
<el-link  
  href="https://element.eleme.cn/#/zh-CN/component/button"  
  target="_blank">Eelement UI</el-link>
```

disabled 设置超链接不可用

```
<el-link disabled  
  href="https://element.eleme.cn/#/zh-CN/component/button"  
  target="_blank">Eelement UI</el-link>
```

underline 设置下划线

```
<el-link
  :underline="false"

href="https://element.eleme.cn/#/zh-
CN/component/button"
  target="_blank">Eelement UI</el-
link>
```

## Element UI

icon 给文字超链接设置图标

```
<el-link
  :underline="false"
  icon="el-icon-phone"

href="https://element.eleme.cn/#/zh-
CN/component/button"
  target="_blank">Eelement UI</el-
link>
```



## 6 Radio 单选框

---

使用 el-radio 标签即可，通过 v-model 进行对象数据的绑定，label 表示该单选框的值，文本直接写入到标签内部即可。

```
<template>
  <el-radio v-model="radio" label="1">选项
1</el-radio>
  <el-radio v-model="radio" label="2">选项
2</el-radio>
</template>

<script>
  data(){
    return{
      underline:true,
      loading:false,
      radio:'2'
    }
  }
</script>
```

change 绑定切换事件

```
<el-radio v-model="radio" label="1"
@change="change">选项1</el-radio>
change(){
  console.log('当前radio的值: '+this.radio)
}
```

## 7 Checkbox 多选框

## 使用 el-checkbox 标签来完成

```
<template>
  <div>
    <el-checkbox
      :indeterminate="isIndeterminate" v-
      model="checkAll"
      @change="handleCheckAllChange">全选</el-
      checkbox>
      <div style="margin: 15px 0;"></div>
      <el-checkbox-group v-
      model="checkedCities"
      @change="handleCheckedCitiesChange">
        <el-checkbox v-for="city in
        cities" :label="city" :key="city">{{city}}
        </el-checkbox>
      </el-checkbox-group>
    </div>
  </template>

<script>
  const cityOptions = ['上海', '北京', '广
  州', '深圳'];
  export default {
    name: "Test",
    methods: {
      test() {
        this.loading = true;
        setTimeout(() => {
          this.loading = false
        })
      }
    }
  }
```



```
        }, 3000)
    },
    change() {
        console.log('当前radio的
值: '+this.radio)
    },
    handleCheckAllChange(val) {
        this.checkedCities = val ?
cityOptions : [];
        this.isIndeterminate =
false;
    },

    handleCheckedCitiesChange(value) {
        let checkedCount =
value.length;
        this.checkAll =
checkedCount === this.cities.length;
        this.isIndeterminate =
checkedCount > 0 && checkedCount <
this.cities.length;
    }
},
data() {
    return {
        underline: true,
        loading: false,
        radio: '2',
        checkAll: false,
```

```

        checkedCities: ['上海', '北
京'],
        cities: cityOptions,
        isIndeterminate: true
    }
}
}
</script>

```

## 8 Input 输入框

```

<template>
  <div>
    <el-input v-model="input"></el-
input>
    <el-button type="primary"
@click="click">主要按钮</el-button>
  </div>
</template>
click(){
  this.input = 'abc'
}

```

通过 size 属性修改输入框的尺寸，  
large/medium/small/mini，size 修改的是输入框的高  
度

```
<el-input v-model="input" size="large">
</el-input>
<div style="margin: 15px 0"></div>
<el-input v-model="input" size="medium">
</el-input>
<div style="margin: 15px 0"></div>
<el-input v-model="input" size="small">
</el-input>
<div style="margin: 15px 0"></div>
<el-input v-model="input" size="mini"></el-input>
<div style="margin: 15px 0"></div>
```

修改宽度可以在外层嵌套一个 div，通过修改 div 的宽度来实现输入框宽度的修改

```
<div style="width: 300px">
  <el-input v-model="input" size="large">
</el-input>
  <div style="margin: 15px 0"></div>
  <el-input v-model="input"
size="medium"></el-input>
  <div style="margin: 15px 0"></div>
  <el-input v-model="input" size="small">
</el-input>
  <div style="margin: 15px 0"></div>
  <el-input v-model="input" size="mini">
</el-input>
  <div style="margin: 15px 0"></div>
</div>
```

show-password 属性设置可以切换显示隐藏的密码框

```
<el-input v-model="input" size="large"
show-password></el-input>
```

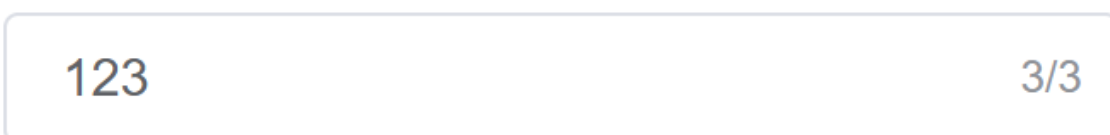
prefix-icon、suffix-icon 属性设置输入框首尾的图标

```
<el-input v-model="input" size="large"
show-password prefix-icon="el-icon-phone"
suffix-icon="el-icon-phone-outline"
"></el-input>
```



maxlength、minlength 限制输入框的字符长度

```
<el-input v-model="input" size="medium"
maxlength="3" show-word-limit></el-input>
```



## 9 Select 下拉框

el-select/el-option 标签进行操作，v-model 进行数据绑定，label 进行文本的展示，value 是当前选项的值。

```
<template>
  <div style="width: 300px">
```

```
      <el-select v-model="value"
placeholder="请选择">
        <el-option
            v-for="item in options"
            :key="item.value"
            :label="item.label"
            :value="item.value">
        </el-option>
    </el-select>
    <el-button @click="click">test</el-
button>
    </div>
</template>
```

```
<script>
    export default {
        name: "Test",
        methods:{
            click(){
                console.log(this.value)
            }
        },
        data(){
            return{
                options: [{
                    value: '选项1',
                    label: '黄金糕'
                }, {
                    value: '选项2',
                    label: '双皮奶'
```

```

        }, {
            value: '选项3',
            label: '蚵仔煎'
        }, {
            value: '选项4',
            label: '龙须面'
        }, {
            value: '选项5',
            label: '北京烤鸭'
        }
    ],
    value: '选项2'
}
}
}
</script>

```

disabled = true, 禁用某个选项

```

<template>
  <div style="width: 300px">
    <el-select v-model="value"
placeholder="请选择">
      <el-option
        v-for="item in options"
        :key="item.value"
        :label="item.label"
        :value="item.value"

        :disabled="item.disabled">
      </el-option>
    </el-select>
  </div>
</template>

```

```
<el-button @click="click">test</el-button>
</div>
</template>
```

```
<script>
  export default {
    name: "Test",
    methods: {
      click() {
        console.log(this.value)
      }
    },
    data() {
      return {
        options: [{
          value: '选项1',
          label: '黄金糕',
          disabled: true
        }, {
          value: '选项2',
          label: '双皮奶'
        }, {
          value: '选项3',
          label: '蚵仔煎',
          disabled: true
        }, {
          value: '选项4',
          label: '龙须面'
        }, {

```



```
        value: '选项5',
        label: '北京烤鸭'
    }],
    value: '选项2'
}
}
}
</script>
```

双皮奶

test

黄金糕

**双皮奶**

蚵仔煎

龙须面

北京烤鸭

change 下拉框进行修改之后会自动触发该事件

```
<template>
```

```

    <div style="width: 300px">
      <el-select v-model="value"
placeholder="请选择" @change="change">
        <el-option
          v-for="item in options"
          :key="item.value"
          :label="item.label"
          :value="item.value"

:disabled="item.disabled">
          </el-option>
        </el-select>
        <el-button @click="click">test</el-
button>
      </div>
</template>

<script>
  export default {
    name: "Test",
    methods: {
      click() {
        console.log(this.value)
      },
      change() {
        console.log('当前选择的
是: '+this.value)
      }
    },
    data() {

```

```
return{
  options: [{
    value: '选项1',
    label: '黄金糕',
    disabled:true
  }, {
    value: '选项2',
    label: '双皮奶'
  }, {
    value: '选项3',
    label: '蚵仔煎',
    disabled:true
  }, {
    value: '选项4',
    label: '龙须面'
  }, {
    value: '选项5',
    label: '北京烤鸭'
  }],
  value: '选项2'
}
}
}
</script>
```

## 10 Switch 开关

---

Switch 组件表示两种相互对立状态之间的切换，开关，`el-switch` 标签完成，`v-model` 进行数据绑定，`boolean`，表示开/关的状态，`active-color` 属性与 `inactive-color` 属性来设置开关的背景色。

```
<el-switch
  v-model="value"
  active-color="#13ce66"
  inactive-color="#ff4949">
</el-switch>
value: true
```

`active-text` 和 `inactive-text` 分别设置开/关对应的文本

```
<el-switch
  v-model="value"
  active-color="#13ce66"
  active-text="上架"
  inactive-color="#ff4949"
  inactive-text="下架">
</el-switch>
value: true
```

`change` 事件进行开/关操作时触发该方法。

```
<el-switch
  v-model="value"
  active-color="#13ce66"
  active-text="上架"
  inactive-color="#ff4949"
  inactive-text="下架"
  @change="change">

</el-switch>

change(){
  console.log('当前开关的状态
是: '+this.value)
}

value: true
```

# 11 Upload 上传文件

---

## 11.1 前端

---

使用 el-upload 组件完成，action 属性为后端请求的接口。

```
<el-upload
  class="upload-demo"
  drag

  action="https://jsonplaceholder.typicode.com/posts/"
  multiple>
  <i class="el-icon-upload"></i>
  <div class="el-upload__text">将文件拖到此
处，或<em>点击上传</em></div>
  <div class="el-upload__tip" slot="tip">
只能上传xlsx文件，且不超过500kb</div>
</el-upload>
```

## 11.2 后端

Spring Boot + EasyExcel 完成 Excel 数据的解析。

1、pom.xml 导入 EasyExcel 相关依赖

```
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>easyexcel</artifactId>
  <version>2.2.6</version>
</dependency>
```

2、创建一个类，映射 Excel 文件。

编号	姓名	性别	年龄	班级
1	张三	男	20	Java1班
2	李四	男	20	Java1班
3	王五	女	20	Java2班

```
package com.southwind.vo;

import
com.alibaba.excel.annotation.ExcelProperty;
import lombok.Data;

@Data
public class ExcelVO {
    @ExcelProperty("编号")
    private Integer id;
    @ExcelProperty("姓名")
    private String name;
    @ExcelProperty("性别")
    private String gender;
    @ExcelProperty("年龄")
    private Integer age;
    @ExcelProperty("班级")
    private String classes;
}
```

3、创建 ExcelService，实现 EasyExcel 的数据解析。

```
package com.southwind.service;

import com.southwind.vo.ExcelVO;

import java.io.InputStream;
import java.util.List;

public interface ExcelService {
    public List<ExcelVO> list(InputStream
inputStream);
}
```

```
package com.southwind.service.impl;

import com.alibaba.excel.EasyExcel;
import
com.alibaba.excel.context.AnalysisContext;
import
com.alibaba.excel.event.AnalysisEventListener;
import com.southwind.service.ExcelService;
import com.southwind.vo.ExcelVO;
import
org.springframework.stereotype.Service;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

@Service
```



```
public class ExcelServiceImpl implements
ExcelService {
    @Override
    public List<ExcelVO> list(InputStream
inputStream) {
        List<ExcelVO> list = new
ArrayList<>();
        EasyExcel.read(inputStream)
            .head(ExcelVO.class)
            .sheet()
            .registerReadListener(new
AnalysisEventListener<ExcelVO>() {

                @Override
                public void
invoke(ExcelVO excelVO, AnalysisContext
analysisContext) {

                    list.add(excelVO);
                }

                @Override
                public void
doAfterAllAnalysed(AnalysisContext
analysisContext) {

                    System.out.println("数据解析完成");
                }
            }).doRead();
        return list;
    }
}
```

```
}
```

4、创建 ExcelController，接收前端请求，调用 ExcelService 进行数据解析。

```
package com.southwind.controller;

import com.southwind.service.ExcelService;
import com.southwind.vo.ExcelVO;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.PostMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;
import
org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.util.List;
```

```
@RestController
@RequestMapping("/excel")
public class ExcelController {

    @Autowired
    private ExcelService excelService;

    @PostMapping("/import")
    public String
importData(@RequestParam("file")MultipartFi
le file){
        try {
            List<ExcelVO> list =
this.excelService.list(file.getInputStream(
));
            for (ExcelVO excelVO : list) {

                System.out.println(excelVO);
            }
        } catch (IOException e) {
            return "fail";
        }
        return "success";
    }
}
```

## 12 Form 表单

---

## 12.1 基本使用

Form 组件，每一个表单域由一个 form-item 组件构成的，表单域中可以放置各种类型的表单控件，Input、Select、Checkbox、Radio、Switch，表单域的值直接跟 Vue 对象进行绑定。

```
<el-form ref="form" :model="form" label-  
width="80px">  
  <el-form-item label="活动名称">  
    <el-input v-model="form.name"></el-  
input>  
  </el-form-item>  
  <el-form-item label="活动区域">  
    <el-select v-model="form.region"  
placeholder="请选择活动区域">  
      <el-option label="区域一"  
value="shanghai"></el-option>  
      <el-option label="区域二"  
value="beijing"></el-option>  
    </el-select>  
  </el-form-item>  
  <el-form-item label="即时配送">  
    <el-switch v-model="form.delivery">  
</el-switch>  
  </el-form-item>  
  <el-form-item label="活动性质">  
    <el-checkbox-group v-  
model="form.type">
```

```
        <el-checkbox label="美食/餐厅线上
活动" name="type"></el-checkbox>
        <el-checkbox label="地推活动"
name="type"></el-checkbox>
        <el-checkbox label="线下主题活动"
name="type"></el-checkbox>
        <el-checkbox label="单纯品牌曝光"
name="type"></el-checkbox>
    </el-checkbox-group>
</el-form-item>
    <el-form-item label="特殊资源">
        <el-radio-group v-
model="form.resource">
            <el-radio label="线上品牌商赞助">
</el-radio>
            <el-radio label="线下场地免费">
</el-radio>
        </el-radio-group>
    </el-form-item>
    <el-form-item>
        <el-button type="primary"
@click="onSubmit">立即创建</el-button>
        <el-button>取消</el-button>
    </el-form-item>
</el-form>
```

活动名称

活动区域

即时配送 ☐

活动性质 ☐ 美食/餐厅线上活动 ☐ 地推活动 ☐ 线下主题活动 ☐ 单纯品牌曝光

特殊资源 ☐ 线上品牌商赞助 ☐ 线下场地免费

## 12.2 数据校验

```
<el-form ref="form" :model="form"
:rules="rules" label-width="80px">
  <el-form-item label="活动名称"
prop="name">
    <el-input v-model="form.name"></el-
input>
  </el-form-item>
  <el-form-item label="活动区域"
prop="region">
    <el-select v-model="form.region"
placeholder="请选择活动区域">
      <el-option label="区域一"
value="shanghai"></el-option>
      <el-option label="区域二"
value="beijing"></el-option>
    </el-select>
  </el-form-item>
```

```
      <el-form-item label="即时配送"
prop="delivery">
        <el-switch v-model="form.delivery">
</el-switch>
      </el-form-item>
      <el-form-item label="活动性质"
prop="type">
        <el-checkbox-group v-
model="form.type">
          <el-checkbox label="美食/餐厅线上
活动" name="type"></el-checkbox>
          <el-checkbox label="地推活动"
name="type"></el-checkbox>
          <el-checkbox label="线下主题活动"
name="type"></el-checkbox>
          <el-checkbox label="单纯品牌曝光"
name="type"></el-checkbox>
        </el-checkbox-group>
      </el-form-item>
      <el-form-item label="特殊资源"
prop="resource">
        <el-radio-group v-
model="form.resource">
          <el-radio label="线上品牌商赞助">
</el-radio>
          <el-radio label="线下场地免费">
</el-radio>
        </el-radio-group>
      </el-form-item>
      <el-form-item>
```

```
      <el-button type="primary"
@click="onSubmit('form')">立即创建</el-
button>
      <el-button>取消</el-button>
    </el-form-item>
  </el-form>
```

```
methods:{
  onSubmit(formName){

    this.$refs[formName].validate((valid) => {
      if (valid) {
        console.log(this.form);
      }
    });
  }
},
data(){
  return{
    form: {
      name: '',
      region: '',
      delivery: false,
      type: [],
      resource: ''
    },
    rules: {
      name: [
        { required: true, message:
'请输入活动名称', trigger: 'blur' },
```



```
        { min: 3, max: 5, message:
'长度在 3 到 5 个字符', trigger: 'blur' }
    ],
    region: [
        { required: true, message:
'请选择活动区域', trigger: 'change' }
    ],
    type: [
        { type: 'array', required:
true, message: '请至少选择一个活动性质',
trigger: 'change' }
    ],
    resource: [
        { required: true, message:
'请选择活动资源', trigger: 'change' }
    ]
}
}
```

## 12.3 自定义数据校验

邮箱: `/^([a-zA-Z0-9-])+@([a-zA-Z0-9-])+(\.[a-zA-Z0-9_-])+/`

```

<el-form ref="form" :model="form"
:rules="rules" label-width="80px">
  <el-form-item label="邮箱" prop="email">
    <el-input v-model="form.email">
  </el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary"
@click="onSubmit('form')">立即创建</el-
button>
    <el-button>取消</el-button>
  </el-form-item>
</el-form>

```

```

methods:{
  onSubmit(formName){

    this.$refs[formName].validate((valid) => {
      if (valid) {
        console.log(this.form);
      }
    });
  },
  data(){
    var checkEmail = (rule,value,callback)
=> {
      const mailReg = /^[a-zA-Z0-
9_-])+@([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+/
      if(!value){

```

```

        return callback(new Error('邮箱不能
为空'))
    }
    setTimeout(()=>{
        if(mailReg.test(value)){
            callback()
        }else{
            callback(new Error('请输入正确
的邮箱格式'))
        }
    },100)
};
return{
    form: {
        email: ''
    },
    rules: {
        email: [
            { required: true,
validator:checkEmail, trigger: 'blur' },
        ]
    }
}
}

```

## 12.4 数字类型校验

```

<el-form ref="form" :model="form" label-
width="80px">
  <el-form-item label="年龄" prop="age"
    :rules="[
      { required: true, message: '年龄
不能为空'},
      { type:'number', message: '年龄
必须为数字值'}
    ]">
    <el-input v-
model.number="form.age"></el-input>
  </el-form-item>
  <el-form-item>
    <el-button type="primary"
@click="onSubmit('form')">立即创建</el-
button>
    <el-button>取消</el-button>
  </el-form-item>
</el-form>

```

```

methods:{
  onSubmit(formName){

    this.$refs[formName].validate((valid) => {
      if (valid) {
        console.log(this.form);
      }
    });
  }
},

```

```

data(){
  var checkEmail = (rule,value,callback)
=> {
    const mailReg = /^[a-zA-Z0-9_-]+@([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+\/
    if(!value){
      return callback(new Error('邮箱不能
为空'))
    }
    setTimeout(()=>{
      if(mailReg.test(value)){
        callback()
      }else{
        callback(new Error('请输入正确
的邮箱格式'))
      }
    },100)
  };
  return{
    form: {
      age: ''
    }
  }
}

```

# 13 CRUD 小案例

## 13.1 后端

## 1、数据库

	bookid	name	price
1	1	Java实战	31.00
2	2	JavaWeb入门和提高	45.00
3	3	我要学Java	40.00
4	4	springboot技术详解	25.00
5	5	spring实战	35.00
6	6	Hibernate原理	65.00

## 2、pom.xml 添加 MyBatis Plus 相关依赖

```
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-
starter</artifactId>
    <version>3.3.2</version>
</dependency>

<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-
generator</artifactId>
    <version>3.3.2</version>
</dependency>

<dependency>
    <groupId>org.apache.velocity</groupId>
    <artifactId>velocity</artifactId>
    <version>1.7</version>
```

```
</dependency>
```

### 3、通过 MyBatis Plus Generator 自动生成后端代码。

```
package com.southwind;

import
com.baomidou.mybatisplus.annotation.DbType;
import
com.baomidou.mybatisplus.generator.AutoGene
rator;
import
com.baomidou.mybatisplus.generator.config.D
ataSourceConfig;
import
com.baomidou.mybatisplus.generator.config.G
lobalConfig;
import
com.baomidou.mybatisplus.generator.config.P
ackageConfig;
import
com.baomidou.mybatisplus.generator.config.S
trategyConfig;
import
com.baomidou.mybatisplus.generator.config.r
ules.NamingStrategy;

public class Main {
    public static void main(String[] args)
    {
        //创建对象
```

```
        AutoGenerator autoGenerator = new
AutoGenerator();
        //数据源
        DataSourceConfig dataSourceConfig =
new DataSourceConfig();

        dataSourceConfig.setDbType(DbType.MYSQL);

        dataSourceConfig.setDriverName("com.mysql.
cj.jdbc.Driver");

        dataSourceConfig.setUrl("jdbc:mysql://loca
lhost:3306/dbname");

        dataSourceConfig.setUsername("root");

        dataSourceConfig.setPassword("root");

        autoGenerator.setDataSource(dataSourceConf
ig);
        //全局配置
        GlobalConfig globalConfig = new
GlobalConfig();

        globalConfig.setOutputDir(System.getProper
ty("user.dir")+"/src/main/java");
        globalConfig.setAuthor("admin");
        globalConfig.setOpen(false);
        //去掉Service的I
```



```
globalConfig.setServiceName("%sService");

autoGenerator.setGlobalConfig(globalConfig
);

    //包配置
    PackageConfig packageConfig = new
PackageConfig();

    packageConfig.setParent("com.southwind");
    packageConfig.setEntity("entity");
    packageConfig.setMapper("mapper");

    packageConfig.setService("service");

    packageConfig.setServiceImpl("service.impl
");

    packageConfig.setController("controller");

    autoGenerator.setPackageInfo(packageConfig
);

    //策略配置
    StrategyConfig strategyConfig = new
StrategyConfig();
    strategyConfig.setInclude("book");

    strategyConfig.setNaming(NamingStrategy.un
derline_to_camel);
```

```
strategyConfig.setColumnNaming(NamingStrategy.underline_to_camel);

strategyConfig.setEntityLombokModel(true);

autoGenerator.setStrategy(strategyConfig);
    //启动
    autoGenerator.execute();
}
}
```

#### 4、Controller 中添加业务代码。

```
package com.southwind.controller;

import com.southwind.entity.Book;
import com.southwind.service.BookService;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.*;

import
org.springframework.stereotype.Controller;

import java.util.List;

/**
```

```
* <p>
*  前端控制器
* </p>
*
* @author admin
* @since 2021-12-26
*/
```

```
@RestController
@RequestMapping("/book")
public class BookController {

    @Autowired
    private BookService bookService;

    @GetMapping("/list")
    public List<Book> list(){
        return this.bookService.list();
    }

    @GetMapping("/findById/{id}")
    public Book
findById(@PathVariable("id") Integer id){
        return
this.bookService.getById(id);
    }

    @DeleteMapping("/delete/{id}")
    public boolean
delete(@PathVariable("id") Integer id){
```

```
        return
this.bookService.removeById(id);
    }

    @PostMapping("/add")
    public boolean add(@RequestBody Book
book){
        return this.bookService.save(book);
    }

    @PutMapping("/update")
    public boolean update(@RequestBody Book
book){
        return
this.bookService.updateById(book);
    }
}
```

5、 application.yml

```
spring:
  datasource:
    driver-class-name:
com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/dbname
    username: root
    password: root
  mybatis:
    configuration:
      log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
      type-aliases-package:
com.southwind.entity
  server:
    port: 8181
```

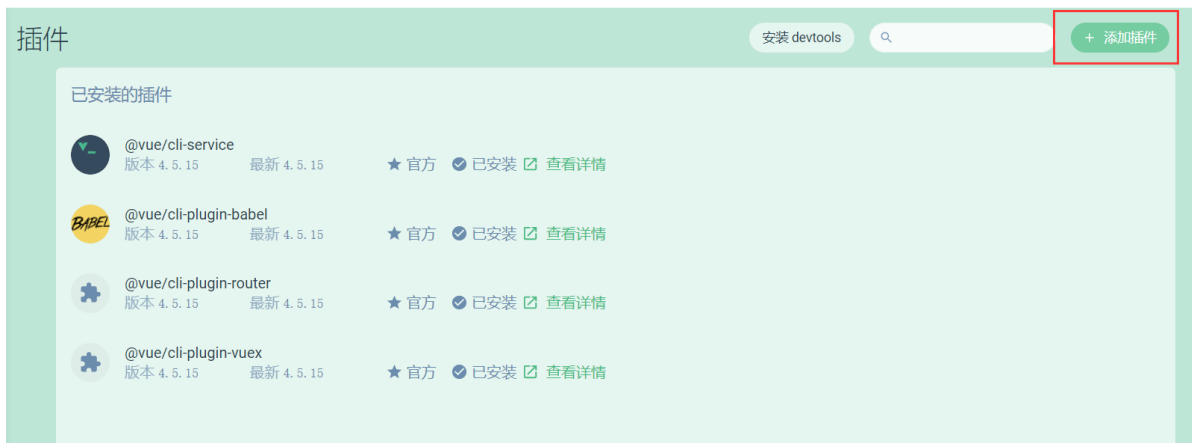
## 13.2 前端

---

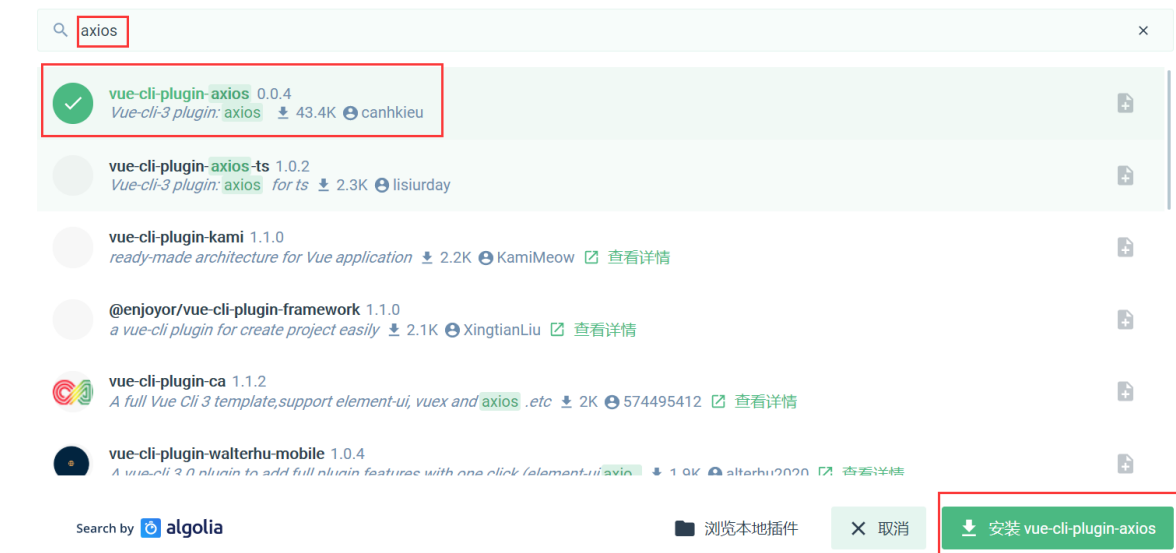
1、安装 axios 插件，点击插件



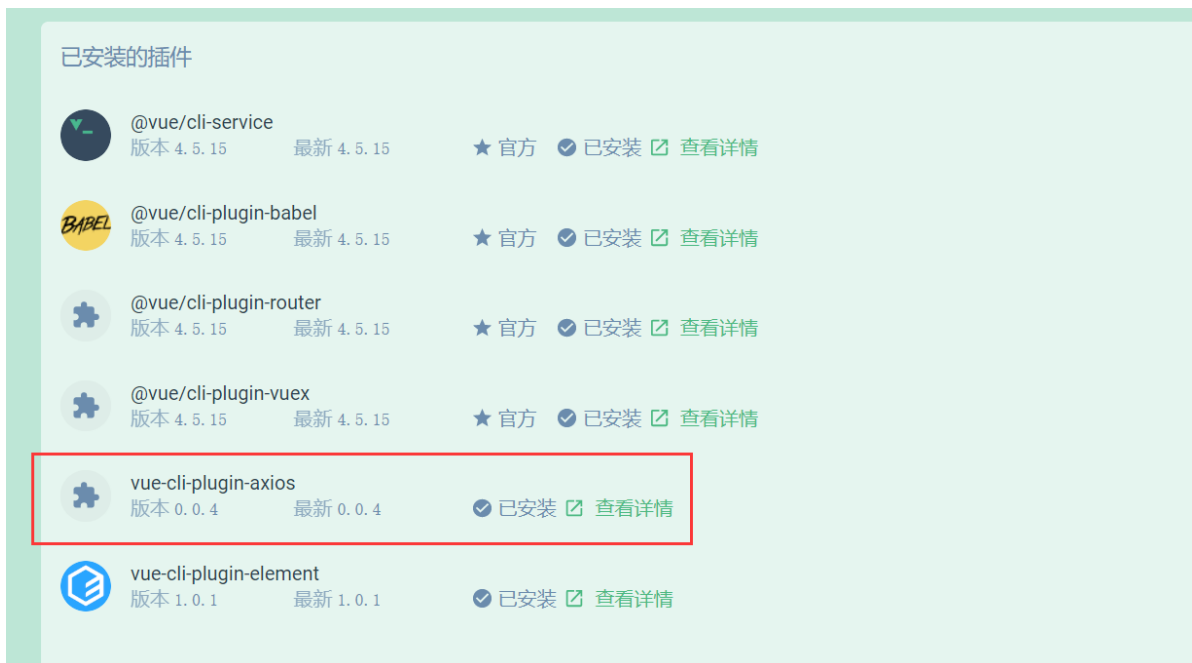
## 2、点击添加插件



## 3、搜索框输入 axios，选择第一项，点击安装按钮



#### 4、安装成功，如下所示。



#### 5、首页数据加载。

```
<template>
  <el-table
    :data="tableData"
    border
    style="width: 800px">
    <el-table-column
      fixed
      prop="bookid"
```

```
        label="编号"
        width="150">
    </el-table-column>
    <el-table-column
        prop="name"
        label="书名"
        width="200">
    </el-table-column>
    <el-table-column
        prop="price"
        label="价格"
        width="150">
    </el-table-column>
    <el-table-column
        fixed="right"
        label="操作"
        width="200">
        <template slot-scope="scope">
            <el-button
                size="mini"

        @click="handleEdit(scope.$index,
scope.row)">编辑</el-button>
            <el-button
                size="mini"
                type="danger"

        @click="handleDelete(scope.$index,
scope.row)">删除</el-button>
        </template>
```



```

        </el-table-column>
    </el-table>
</template>

<script>
    export default {
        name: "Index",
        created() {
            let _this = this

            axios.get('http://localhost:8181/book/list')
                .then(function (resp) {
                    _this.tableData = resp.data
                })
        },
        data() {
            return {
                tableData: ''
            }
        }
    }
</script>

```

## 6、删除数据。

```

<template>
    <el-table
        :data="tableData"
        border
        style="width: 800px">
        <el-table-column

```

```
        fixed
        prop="bookid"
        label="编号"
        width="150">
    </el-table-column>
    <el-table-column
        prop="name"
        label="书名"
        width="200">
    </el-table-column>
    <el-table-column
        prop="price"
        label="价格"
        width="150">
    </el-table-column>
    <el-table-column
        fixed="right"
        label="操作"
        width="200">
        <template slot-scope="scope">
            <el-button
                size="mini"
                @click="handleEdit(
scope.row)">编辑</el-button>
            <el-button
                size="mini"
                type="danger"
                @click="handleDelete(scope.row)">删除</el-
button>
```

```

        </template>
      </el-table-column>
    </el-table>
  </template>

  <script>
    export default {
      name: "Index",
      methods: {
        handleDelete(row) {
          let _this = this
          this.$confirm('是否确定删除
《'+row.name+'》?', '提示', {
            confirmButtonText: '确
定',
            cancelButtonText: '取
消',
            type: 'warning'
          }).then(() => {

            axios.delete('http://localhost:8181/book/d
elete/'+row.bookid).then(function (resp) {
              if(resp.data){

                _this.$alert('《'+row.name+'》删除成功', '提
示', {

                  confirmButtonText: '确定',
                  callback:
                    action => {

```

```

location.reload()

    }
    });
}
})
}).catch(() => {

});
}
},
created() {
    let _this = this

    axios.get('http://localhost:8181/book/list')
        .then(function (resp) {
            _this.tableData = resp.data
        })
    },
    data() {
        return {
            tableData: ''
        }
    }
}
</script>

```

7、修改数据。

Index

```
<template>
  <el-table
    :data="tableData"
    border
    style="width: 800px">
    <el-table-column
      fixed
      prop="bookid"
      label="编号"
      width="150">
    </el-table-column>
    <el-table-column
      prop="name"
      label="书名"
      width="200">
    </el-table-column>
    <el-table-column
      prop="price"
      label="价格"
      width="150">
    </el-table-column>
    <el-table-column
      fixed="right"
      label="操作"
      width="200">
      <template slot-scope="scope">
        <el-button
          size="mini"
          @click="handleEdit(
scope.row)">编辑</el-button>
```

```

        <el-button
            size="mini"
            type="danger"

            @click="handleDelete(scope.row)">删除</el-
button>

        </template>
    </el-table-column>
</el-table>
</template>

<script>
    export default {
        name: "Index",
        methods: {
            handleEdit(row) {
                this.$router.push('/update?
id='+row.bookid)
            },
            handleDelete(row) {
                let _this = this
                this.$confirm('是否确定删除
《'+row.name+'》?', '提示', {
                    confirmButtonText: '确
定',
                    cancelButtonText: '取
消',
                    type: 'warning'
                }).then(() => {

```

```

        axios.delete('http://localhost:8181/book/delete/'+row.bookid).then(function (resp) {
            if(resp.data){

                _this.$alert('《'+row.name+'》删除成功', '提示', {

                    confirmButtonText: '确定',
                    callback:
                    action => {

                        location.reload()

                    }
                });
            }
        })
    }).catch(() => {

    });
},
created() {
    let _this = this

    axios.get('http://localhost:8181/book/list').then(function (resp) {
        _this.tableData = resp.data
    })
},

```

```

        data() {
            return {
                tableData: ''
            }
        }
    }
</script>

```

Update

```

<template>
  <div style="width: 300px">
    <el-form ref="form" :model="form"
:rules="rules" label-width="80px">
      <el-form-item label="编号"
prop="bookid" :rules="[
        { required: true,
message: '编号不能为空'},
        { type:'number',
message: '编号必须为数字值'}
      ]">
        <el-input v-
model.number="form.bookid" readonly></el-
input>
      </el-form-item>
      <el-form-item label="书名"
prop="name">
        <el-input v-
model="form.name"></el-input>
      </el-form-item>
    </el-form>
  </div>
</template>

```



```

        <el-form-item label="价格"
prop="price" :rules="[
            { required: true,
message: '价格不能为空'},
            { type:'number',
message: '价格必须为数字值'}
        ]">
            <el-input v-
model.number="form.price"></el-input>
        </el-form-item>
        <el-form-item>
            <el-button type="primary"
@click="onSubmit('form')">立即修改</el-
button>
            <el-button>取消</el-button>
        </el-form-item>
    </el-form>
</div>
</template>

<script>
    export default {
        name: "Update",
        created() {
            let _this = this

            axios.get('http://localhost:8181/book/find
ById/'+this.$route.query.id).then(function
(resp) {
                _this.form = resp.data
            })
        }
    }

```

```

        })
    },
    data(){
        return{
            form: {
                bookid: '',
                name: '',
                price: ''
            },
            rules: {
                name: [
                    { required: true,
message: '请输入图书名称', trigger: 'blur' },
                    { min: 3, max: 20,
message: '长度在 3 到 20 个字符', trigger:
'blur' }
                ]
            }
        },
        methods:{
            onSubmit(formName) {
                let _this = this

                this.$refs[formName].validate((valid) => {
                    if (valid) {

                        axios.put('http://localhost:8181/book/update',this.form).then(function (resp) {
                            if(resp.data){

```

```

    _this.$alert('《'+_this.form.name+'》修改成功', '', {
      confirmButtonText: '确定',
      callback: action => {
        _this.$router.push('/index')
      }
    });
  }
}
}
}
</script>

```

## 8、添加数据。

```

<template>
  <div style="width: 300px">
    <el-form ref="form" :model="form"
    :rules="rules" label-width="80px">
      <el-form-item label="编号"
      prop="bookid" :rules="[
        { required: true,
        message: '编号不能为空'},

```

```
                { type: 'number',
message: ' 编号必须为数字值' }
            ]">
            <el-input v-
model.number="form.bookid"></el-input>
        </el-form-item>
        <el-form-item label="书名"
prop="name">
            <el-input v-
model="form.name"></el-input>
        </el-form-item>
        <el-form-item label="价格"
prop="price" :rules="[
                { required: true,
message: ' 价格不能为空' },
                { type: 'number',
message: ' 价格必须为数字值' }
            ]">
            <el-input v-
model.number="form.price"></el-input>
        </el-form-item>
        <el-form-item>
            <el-button type="primary"
@click="onSubmit('form')">立即修改</el-
button>
            <el-button>取消</el-button>
        </el-form-item>
    </el-form>
</div>
</template>
```

```
<script>
  export default {
    name: "Add",
    data(){
      return{
        form: {
          bookid: '',
          name: '',
          price: ''
        },
        rules: {
          name: [
            { required: true,
message: '请输入图书名称', trigger: 'blur' },
            { min: 3, max: 20,
message: '长度在 3 到 20 个字符', trigger:
'blur' }
          ]
        }
      },
      methods:{
        onSubmit(formName) {
          let _this = this

this.$refs[formName].validate((valid) => {
          if (valid) {
```

```
    axios.post('http://localhost:8181/book/add',this.form).then(function (resp) {
        if(resp.data){

            _this.$alert('《'+_this.form.name+'》添加成功', '', {

                confirmButtonText: '确定',

                callback: action => {

                    _this.$router.push('/index')

                }

            });

        }

    });

}

}

}

</script>
```