

十次方前端系统开发day03

学习目标：

- 掌握elementUI提供的脚手架搭建管理后台的方法
- 掌握elementUI的table组件的使用，实现列表功能
- 掌握elementUI的pagination组件的使用，实现分页功能
- 掌握elementUI的form相关组件的使用，实现条件查询功能
- 掌握elementUI的dialog组件和\$message的使用，实现弹出窗口和消息提示功能
- 掌握elementUI的select组件的使用，实现下拉列表功能
- 实现新增数据和修改数据的功能
- 掌握confirm的使用，实现询问框，实现删除功能

第一章-管理后台搭建

我们的十次方管理后台就采用ElementUI来进行搭建.

1. 什么是ElementUI

Element 饿了么为开发者准备的基于 Vue 的桌面端组件库

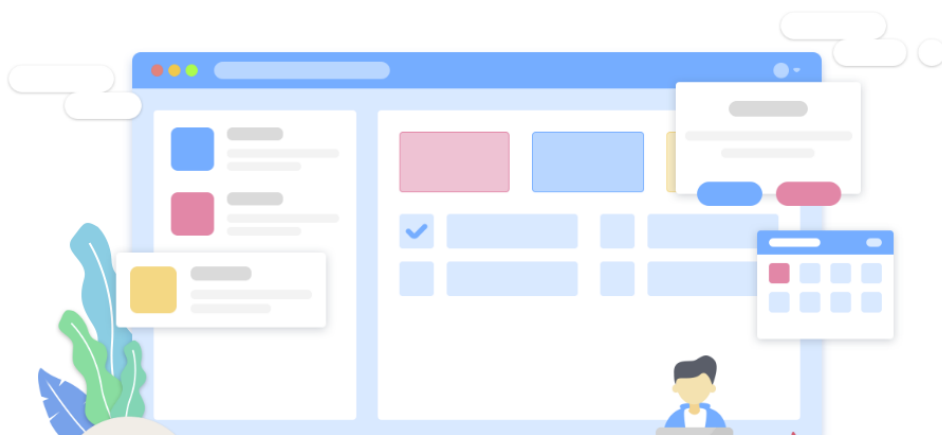
官网：<http://element-cn.eleme.io/#/zh-CN>



指南 组件 资源 中文 ▾

网站快速成型工具

Element，一套为开发者、设计师和产品经理准备的基于 Vue 2.0 的桌面端组件库



2. 脚手架

2.1 什么是脚手架

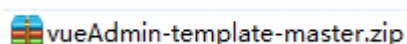
软件开发的脚手架的概念是从建筑学术语上引申而来。在建筑学上，脚手架指施工现场为工人操作并解决垂直和水平运输而搭设的各种支架。你见过大楼施工吧？在大楼外墙围的那圈支架就是其中一种脚手架。在软件开发上（当然也包括前端开发）的脚手架指的就是：有人帮你把这个开发过程中要用到的工具、环境都配置好了，你就可以方便地直接开始做开发，专注你的业务，而不用再花时间去配置这个开发环境，这个开发环境就是脚手架。



2.2 快速搭建

官网上提供了非常基础的脚手架，如果我们使用官网的脚手架需要自己写很多代码比如登陆界面、主界面菜单等内容。课程已经提供了功能完整的脚手架，我们可以拿过来在此基础上开发，这样可以极大节省我们开发的时间。

(1) 解压vueAdmin-template-master



(2) 在命令提示符进入该目录，输入命令：

```
npm install
```

这样下载安装所有的依赖，几分钟后下载完成。

(3) 输入命令运行：

```
npm run dev
```

运行后自动弹出浏览器。

2.3 了解工程结构

以下是主要的目录结构：

目录名称	存储内容
build	构建工程相关脚本
config	配置相关
src	工程源码
static	静态资源
src/api	访问后端API
src/utlis	工具类
src/views	页面
src/router	路由
src/assets	静态资源
src/store	状态管理
src/main.js	入口文件

- src/assets和static区别

assets里面的会被webpack打包进你的代码里，而static里面的，就直接引用了。一般在static里面放一些类库的文件，在assets里面放属于该项目的资源文件。

简单的讲，static放别人家的，assets放自己写的。

3.项目初始化

3.1 关闭语法规范性检查

修改config/index.js，将useEslint的值改为false。

```
23 // Use Eslint Loader?
24 // If true, your code will be linted during bu
25 // linting errors and warnings will be shown i
26 useEslint: false,
27 // If true, eslint errors and warnings will al
28 // in the browser.
29 showEslintErrorsInOverlay: false,
30
```

此配置作用: 是否开启语法检查，语法检查是通过ESLint 来实现的。

我们现在科普一下,什么是ESLint: ESLint 是一个语法规则和代码风格的检查工具，可以用来保证写出语法正确、风格统一的代码。如果我们开启了Eslint，也就意味着要接受它非常苛刻的语法检查，包括空格不能少些或多些，必须单引不能双引，语句后不可以写分号等等，这些规则其实是可以设置的。我们作为前端的初学者，最好先关闭这种校验，否则会浪费很多精力在语法的规范性上。如果以后做真正的企业级开发，建议开启。

3.2 国际化设置

打开src/main.js 找到这句代码

```
import locale from 'element-ui/lib/locale/lang/en'
```

我们将en修改为zn-CN

```
import locale from 'element-ui/lib/locale/lang/zh-CN'
```

修改后组件都是按照中文的习惯展示

3.3 与easy-mock对接

启动

```
npm run start
```

(1) 修改config下的dev.env.js中的BASE_API为easy-mock的Base URL

```
BASE_API: '" http://192.168.147.134:7300/mock/5bdbd91b4bf4f209c88e032b"'
```

(2) easy-mock添加登陆认证模拟数据

地址 : /user/login

提交方式 : post

内容 :

```
{
  "code": 20000,
  "data": {
    "token": "admin"
  }
}
```

(3) 添加返回用户信息url模拟数据

地址 : /user/info

提交方式 : get

内容 :

```
{
  "code": 20000,
  "data": {
    "roles": ["admin"],
    "name": "admin",
    "avatar": "https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif"
  }
}
```

(4) 添加退出登录url

地址: /user/logout

提交方式: post

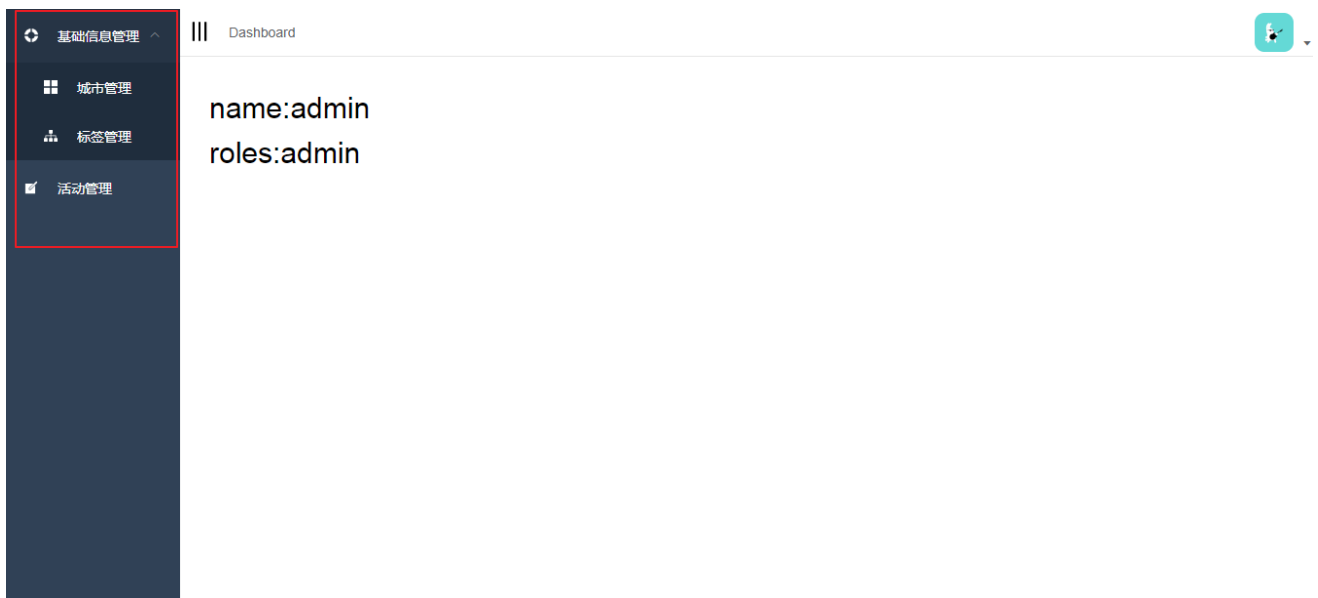
内容:

```
{
  "code": 20000,
  "data": {
  }
}
```

3.4 更改标题与菜单

(1) 修改view/login/index.vue里的标题为"十次方后台管理系统", 修改后浏览器自动刷新。这就是脚手架中已经为我们添加了热部署功能。

(2) 修改src/router 下的index.js 中constantRouterMap的内容



```
export const constantRouterMap = [
  { path: '/login', component: () => import('@/views/login/index'), hidden: true },
  { path: '/404', component: () => import('@/views/404'), hidden: true },
  {
    path: '/',
```

```

    component: Layout,
    redirect: '/dashboard',
    name: 'Dashboard',
    hidden: true,
    children: [{
      path: 'dashboard',
      component: () => import('@/views/dashboard/index')
    }]
  },
  {
    path: '/example',
    component: Layout,
    redirect: '/example/table',
    name: 'Example',
    meta: { title: '基本信息管理', icon: 'example' },
    children: [
      {
        path: 'table',
        name: 'Table',
        component: () => import('@/views/table/index'),
        meta: { title: '城市管理', icon: 'table' }
      },
      {
        path: 'tree',
        name: 'Tree',
        component: () => import('@/views/tree/index'),
        meta: { title: '标签管理', icon: 'tree' }
      }
    ]
  },
  {
    path: '/form',
    component: Layout,
    name: 'Example2',
    meta: { title: '活动管理', icon: 'example' },
    children: [
      {
        path: 'index',
        name: 'Form',
        component: () => import('@/views/form/index'),
        meta: { title: '活动管理', icon: 'form' }
      }
    ]
  },
  { path: '*', redirect: '/404', hidden: true }
]

```

4.项目解读

4.1编写代码的步骤

- 在API目录下编写API请求服务器
- 在views目录下的.vue页面调用API, 获得数据 填充页面

4.2引入API的两种方式

方式一

- API定义

```
import request from '@utils/request'

export function getList(params) {
  return request({
    url: '/table/list', //请求路径
    method: 'get', //请求方式
    data:params //请求参数
  })
}
```

- 在view中引入调用

```
import { getList } from '@api/table'

//调用
getList()
```

方式二

- API定义

```
import request from '@utils/request'

export default{
  getList:function(params) {
    return request({
      url: '/table/list',
      method: 'get',
      data:params
    })
  }
}
```

- 在view中引入调用

```
import tabAPI from '@api/table'

//调用
tabAPI.getList()
```

第二章-活动管理-列表

1.需求分析

实现活动管理的列表页，包括分页，条件查询。

2.表格组件

2.1需求和组件介绍

- 我们这一环节先实现一个简单的列表，如下图所示：

id	活动名称	活动地址	开始时间	结束时间
lbAiAC	身得局展风得	新疆维吾尔自治区 哈密地区 其它区	1991-05-24	1991-05-24
QgIjz	江常内见	台湾 彰化县 员林镇	1971-11-15	1971-11-15
6vaA	而任叫看需	内蒙古自治区 鄂尔多斯市 鄂托克旗	2000-09-30	2000-09-30
CrG	就越派转实	山东省 日照市 五莲县	2008-04-13	2008-04-13
vBIM*2	从出带更记	天津 天津市 河北区	1994-09-03	1994-09-03
Cm\$N	军地数如南几	吉林省 延边朝鲜族自治州 延吉市	1978-09-10	1978-09-10
MUHH	得基五北般给	山西省 忻州市 定襄县	1972-11-11	1972-11-11
@NclwA	什金美南很	山西省 朔州市 山阴县	2007-11-07	2007-11-07
l\$X8o	王东属命别美法民	山西省 运城市 其它区	2014-11-14	2014-11-14
6Sse	细斯门意角天制严	广东省 湛江市 麻章区	2009-10-25	2009-10-25

- 组件介绍

table组件的属性

参数	说明	类型	可选值	默认值
data	显示的数据	array	—	—

table-column组件的属性





参数	说明	类型	可选值	默认值
label	显示的标题	string	—	—
prop	对应列内容的字段名，也可以使用 property 属性	string	—	—
width	对应列的宽度	string	—	—

以上属性为我们代码中使用到的属性，其他属性请查阅官方文档.

<http://element-cn.eleme.io/#/zh-CN/component/table>

2.2准备工作

我们将swaggerAPI同步到easyMock 然后修改/gathering （ GET方法 ）的内容

>	<input type="checkbox"/>	GET	/gathering/gathering	活动全部列表	   
---	--------------------------	-----	----------------------	--------	---


```

{
  "code": 20000,
  "flag": true,
  "message": "查询成功",
  "data|10": [{
    "id": "@string",
    "name": "@cword(4,8)",
    "summary": "@cword(10,15)",
    "detail": "@cword(20,30)",
    "sponsor": "@string",
    "image": "@image",
    "starttime": "@date",
    "endtime": "@date",
    "address": "@county(true)",
    "enrolltime": "@date",
    "state": "1",
    "city": "@city(true)"
  }]
}

```

2.3代码实现

(1) 在src/api创建gathering.js

```

import request from '@utils/request'

export default {
  getList() {
    return request({
      url: '/gathering/gathering',
      method: 'get'
    })
  }
}

```

(2) 在src/views/tables创建gathering.vue中 , 编写脚本部分

```

<template>
</template>
<script>
import gatheringApi from "@api/gathering";

export default {
  //钩子函数
  created() {
    this.fetchData();
  },

  //数据
  data() {
    return {

```

```

        list: []
      };
    },

    //方法
    methods: {
      //抓取数据
      fetchData() {
        gatheringApi.getList().then(response => {
          this.list = response.data;
        });
      }
    }
  };
</script>

```

(3) 修改gathering.vue , 编写html代码部分

```

<template>
  <el-table
    :data="list"
    stripe
    style="width: 100%">
    <el-table-column
      prop="id"
      label="id"
      width="180">
    </el-table-column>
    <el-table-column
      prop="name"
      label="活动名称"
      width="180">
    </el-table-column>
    <el-table-column
      prop="address"
      label="活动地址">
    </el-table-column>
    <el-table-column
      prop="starttime"
      label="开始时间">
    </el-table-column>
    <el-table-column
      prop="starttime"
      label="结束时间">
    </el-table-column>
  </el-table>
</template>

```

(4) 修改src/router/index.js , 修改路由规则

```
62 {
63   path: '/form',
64   component: Layout,
65   children: [
66     {
67       path: 'index',
68       name: 'Form',
69       component: () => import('@views/table/gathering'),
70       meta: { title: '活动管理', icon: 'form' }
71     }
72   ]
73 },
```

3. 分页组件

3.1需求 and 组件介绍

- 需求:我们已经通过表格组件完成了列表的展示，接下来需要使用分页组件完成分页功能



- 分页组件

pagination的常用属性：





参数	说明	类型	可选值	默认值
page-size	每页显示条目个数	Number	—	10
total	总条目数	Number	—	—
current-page	当前页数，支持.sync 修饰符	Number	—	1
layout	组件布局，子组件名用逗号分隔	String	sizes, prev, pager, next, jumper, ->, total, slot	'prev, pager, next, jumper, ->, total'
page-sizes	每页显示个数选择器的选项设置	Number[]	—	[10, 20, 30, 40, 50, 100]

pagination的常用事件：

事件名称	说明	回调参数
size-change	pageSize 改变时会触发	每页条数
current-change	currentPage 改变时会触发	当前页

更多属性方法事件请查看官方文档：<http://element-cn.eleme.io/#/zh-CN/component/pagination>

3.2准备工作

>	<input type="checkbox"/>	POST	/gathering/gathering/search/{page}/{size}	活动分页	   
---	--------------------------	------	---	------	---

修改接口/gathering/search/{page}/{size} method:POST

```
{
  "code": 20000,
  "flag": true,
  "message": "查询成功",
  "data": {
    "total": "@integer(100, 200)",
    "rows|10": [{
      "id": "@string",
      "name": "@cword(4,8)",
      "summary": "@cword(10,15)",
      "detail": "@cword(20,30)",
      "sponsor": "@string",
      "image": "@image",
      "starttime": "@date",
      "endtime": "@date",
      "address": "@county(true)",
      "enrolltime": "@date",
      "state": "1",
      "city": "@city(true)"
    }]
  }
}
```

3.3代码实现

(1) 修改src/api/gathering.js，增加方法导出

```
//分页获得获得列表
search(page,size,searchMap){
  return request({
    url:`/gathering/search/${page}/${size}`,
    method:'post',
    data:searchMap
  })
}
```

(2) 修改src/views/table/gathering.vue , 编写脚本部分

```
<script>
import gatheringApi from "@/api/gathering";

export default {
  //钩子函数
  created() {
    this.fetchData();
  },

  //数据
  data() {
    return {
      list: [],
      total: 0, //总记录数
      curPage: 1, //当前页码
      pageSize: 10, //一页显示的数量
      searchMap: { state: "1" } //查询条件
    };
  },

  //方法
  methods: {
    fetchData() {
      /* return gatheringApi.getList().then(response => {
        this.list = response.data;
      }); */
      gatheringApi
        .serach(this.curPage, this.pageSize, this.searchMap)
        .then(response => {
          this.list = response.data.rows
          this.total = response.data.total
        })
    }
  }
};
</script>
```

(3) 修改src/views/table/gathering.vue , 增加分页栏

```
<template>
<div>
  ...
  <el-pagination
    @size-change="fetchData"
    @current-change="fetchData"
    :current-page="curPage"
    :page-sizes="[5, 10, 20,50]"
    :page-size="pageSize"
    layout="total, sizes, prev, pager, next, jumper"
    :total="total">
```

```

    </el-pagination>
  </div>
</template>

```

注意：template里面要求必须有唯一的跟节点，我们这里用div将表格和分页控件包起来。

4.条件查询

4.1需求和组件介绍

- 在分页列表的基础上实现条件查询功能

活动名称
活动日期

~

id	活动名称	活动地址	开始时间	结束时间
O9Ly*	志计强油	内蒙古自治区 乌海市 海南区	1990-07-26	1990-07-26

- 表单组件:

行内表单

当垂直方向空间受限且表单较简单时，可以在一行内放置表单。

审批人
活动区域

form（表单）组件属性详见官方文档：<http://element-cn.eleme.io/#/zh-CN/component/form>

input（文本框）组件属性详见官方文档：<http://element-cn.eleme.io/#/zh-CN/component/input>

date-picker（日期框）组件属性详见官方文档：<http://element-cn.eleme.io/#/zh-CN/component/date-picker>

4.2代码实现

修改src/views/table/gathering.vue，增加查询表单

```

<br/>
<!-- 表单 -->
<el-form :inline="true" class="demo-form-inline">
  <el-form-item label="活动名称">
    <el-input v-model="searchMap.name" placeholder="获得名称"></el-input>
  </el-form-item>
  <el-form-item label="活动日期">
    <el-date-picker v-model="searchMap.starttime_low" placeholder="开始时间"></el-
date-picker>
    ~
    <el-date-picker v-model="searchMap.starttime_high" placeholder="开始时间"></el-
date-picker>
  </el-form-item>
  <el-form-item>
    <el-button type="primary" @click="fetchData()">查询</el-button>
  </el-form-item>
</el-form>

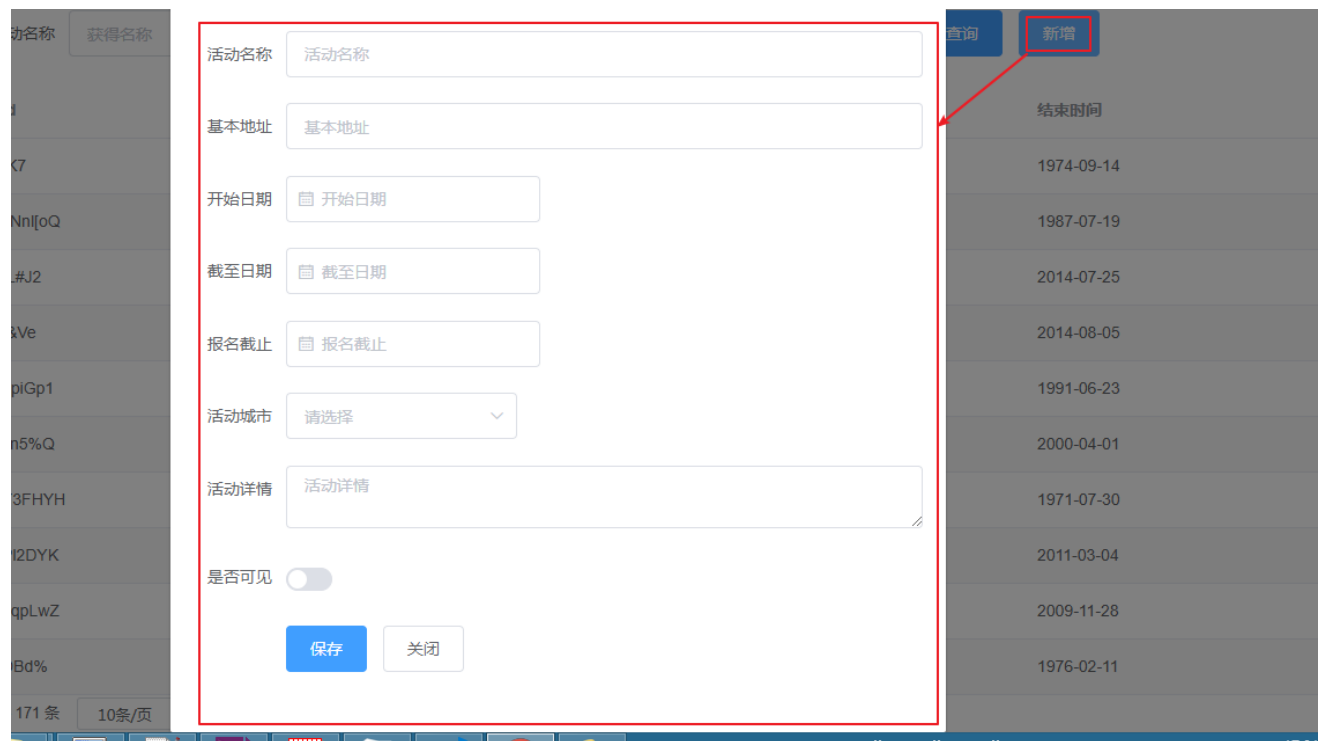
```

```
</el-form-item>
</el-form>
```

第三章-活动管理-增加

1.需求分析

界面中加入"新增"按钮，点击弹出编辑窗口



点击“修改”按钮，关闭窗口并刷新表格，弹出提示（成功或失败）

2.对话框dialog

（1）修改src/api/gathering.js，在template中增加对话框组件

```
<!-- 弹出窗口 -->
<el-dialog title="新增活动" :visible.sync="dialogFormVisible">

</el-dialog>
```

属性title为对话框标题，visible为是否显示。

（2）变量dialogFormVisible用于控制对话框的显示。我们在脚本代码中定义

```

data(){
  return {
    ....
    dialogFormVisible: false //对话框是否显示
  }
}

```

(3) template中增加按钮，用于打开对话框

```

<el-button type="primary" @click="dialogFormVisible=true">新增</el-button>

```

dialog属性详见官方文档：<http://element-cn.eleme.io/#/zh-CN/component/dialog>

3.编辑表单

修改src/views/table/gathering.vue，在弹出窗口添加编辑表单

```

<el-dialog title="编辑" :visible.sync="dialogFormVisible">
  <el-form label-width="80px">
    <el-form-item label="活动名称">
      <el-input v-model="pojo.name" placeholder="活动名称"></el-input>
    </el-form-item>
    <el-form-item label="基本地址">
      <el-input v-model="pojo.address" placeholder="基本地址"></el-input>
    </el-form-item>
    <el-form-item label="开始日期">
      <el-date-picker type="date" v-model="pojo.starttime" placeholder="开始日期"></el-date-
picker>
    </el-form-item>
    <el-form-item label="截至日期">
      <el-date-picker type="date" v-model="pojo.endtime" placeholder="截至日期"></el-date-
picker>
    </el-form-item>
    <el-form-item label="报名截止">
      <el-date-picker type="date" v-model="pojo.enrolltime" placeholder="报名截止"></el-
date-picker>
    </el-form-item>
    <el-form-item label="活动详情">
      <el-input v-model="pojo.detail" placeholder="活动详情" type="textarea" :rows="2"></el-
input>
    </el-form-item>
    <el-form-item label="是否可见">
      <el-switch active-value="1" inactive-value="0" v-model="pojo.status"></el-switch>
    </el-form-item>
    <el-form-item>
      <el-button type="primary" >保存</el-button>
      <el-button @click="dialogFormVisible = false" >关闭</el-button>
    </el-form-item>
  </el-form>
</el-dialog>

```


这里我们主要要掌握多行文本编辑框与开关组件switch的使用

4.下拉选择框

4.1需求和组件介绍

- 需求:在新增窗口实现城市下拉选择框

The screenshot shows a form with several input fields. A dropdown menu is open for the '活动城市' (Activity City) field, displaying a list of cities: 通化市, 秦皇岛市 (highlighted in blue), 鸡西市, 那曲地区, 兰州市 (highlighted in grey), 通化市, and 马鞍山市. The '活动城市' field itself is highlighted with a red rectangle and contains '秦皇岛市'.

- 组件介绍: 我们这里需要使用elementUI提供的下拉选择框select
- ### 基础用法

适用广泛的基础单选

The screenshot shows the Element UI select component. The dropdown menu is open, showing a list of food items: 黄金糕, 双皮奶, 蚬仔煎, 龙须面, and 北京烤鸭. The '请选择' (Please select) placeholder is visible in the input field, which is highlighted with a red rectangle.

4.2准备工作

The screenshot shows the API client interface. The URL bar contains '/base/city' and the method is set to 'GET'. The response is '返回城市列表' (Return city list). The interface is highlighted with a red rectangle.

```

{
  "flag": true,
  "code": 20000,
  "message": "查询成功",
  "data|8": [{
    "id": "@string",
    "name": "@city()",
    "ishot": "1"
  }]
}

```

4.3代码实现

(1) 创建src/api/city.js

```

import request from '@utils/request'

export default{
  //获得城市列表
  getList(){
    return request({
      url:'/city',
      method:'get'
    })
  }
}

```

(2) 修改src/views/table/gathering.vue的js脚本部分

为data添加属性

```
cityList: []
```

引入城市API

```
import cityAPI from '@api/city';
```

修改created，增加对城市方法的调用

```

//钩子函数,等页面加载就执行
created() {
  this.fetchData();
  this.loadCityData();
},

//方法
methods: {
  fetchData() {

```

```

//加载城市数据
cityApi.getList().then(response => {
  this.cityList = response.data;
});
}
}

```

(3) 修改src/views/table/gathering.vue , 增加城市下拉框

```

<el-form-item label="活动城市">
  <el-select v-model="pojo.city" placeholder="请选择">
    <el-option
      v-for="item in cityList"
      :key="item.id"
      :label="item.name"
      :value="item.id">
    </el-option>
  </el-select>
</el-form-item>

```

5.表单提交

(1) 修改easymock中的/gathering (增加活动 POST)

>	<input type="checkbox"/>	POST	/gathering/gathering	增加活动	   
---	--------------------------	------	----------------------	------	---

```

{
  "flag": true,
  "code": 20000,
  'message': "增加成功"
}

```

(2) 修改src/api/gathering.js , 增加方法导出

```

//添加活动
save(pojo){
  return request({
    url:`/gathering`,
    method:'post',
    data:pojo
  })
}

```

(3) 修改src/views/table/gathering.vue的js脚本部分 增加方法执行保存

```
//3.增加活动
handleSave(){
    gatheringAPI.save(this.pojo).then(response=>{
        alert(response.message);
        if(response.flag){
            this.fetchData(); //如果增加成功，重新加载数据
        }
        this.dialogFormVisible = false; //关闭dialog
    });
}
```

(4) 修改弹出框中的“保存”按钮，调用保存方法

```
<el-button type="primary" @click="handleSave()">保存</el-button>
```

第四章-活动管理-修改

1 需求分析

在表格的操作列增加“修改”按钮，点击修改按钮弹出窗口并显示数据，点击保存按钮保存修改并刷新表格。

2 根据ID加载数据

2.1 准备工作

>	<input type="checkbox"/>	GET	/gathering/gathering/{gatheringId}	根据ID查询活动	   
---	--------------------------	-----	------------------------------------	----------	---

修改easymock 接口 /gathering/gathering/{id} (GET)

```
{
  "flag": true,
  "code": 20000,
  'message': "查询成功",
  'data': {
    "id": "1",
    "name": "测试活动",
    "sponsor": "主办方",
    "image": "@image",
    "starttime": "@date",
    "endtime": "@date",
    "address": "@county(true)",
    "enrolltime": "@date",
    "detail": "@cword(40)",
    "status": "1",
    "city": "@string"
  }
}
```

2.2 代码实现

(1) 在表格table中增加模板列,模板列中防止修改按钮,调用handleEdit方法

```
<el-table-column
    fixed="right"
    label="操作"
    width="100">
  <template slot-scope="scope">
    <el-button @click="handleEdit(scope.row.id)" type="text" size="small">更新</el-button>
    <el-button type="text" size="small">删除</el-button>
  </template>
</el-table-column>
```

fixed="right"的作用是定义此列为右固定列

slot-scope用于指定当前行的上下文。使用scope.row可以获取行对象

(2) 修改src/api/gathering.js, 增加方法定义

```
//根据id查询
findById(id){
  return request({
    url: `/gathering/${id}`,
    method: 'get'
  });
}
```

(3) 修改src/views/table/gathering.vue的js脚本部分

新增handleEdit方法

```
//4. 更新
//4.1编辑
handleEdit(id){
  gatheringAPI.findById(id).then(response=>{
    this.pojo = response.data; //回显数据
    this.dialogFormVisible = true; //打开dialog
  });
}
```

3 新增窗口表单清空

测试：我们在点开修改后,关闭窗口,再次新增打开窗口,会发现表单里依然有数据。这样显然是不行的。所以我们要在点击新增时清空表单。这个逻辑我们我们在handleEdit方法中实现

```
//4.更新
//4.1编辑
handleEdit(id) {
  this.dialogFormVisible = true; //打开dialog

  if (id !== "") {
    gatheringAPI.findById(id).then(response => {
      this.pojo = response.data; //回显数据
    });
  } else {
    this.pojo = {};
  }
}
```

修改新增按钮，调用handleEdit方法时传递空字符串

```
<el-button type="primary" @click="handleEdit('')">新增</el-button>
```

4 保存修改

4.1准备工作

修改easymock 接口 /gathering/gathering/{id} （ PUT ）



```
{
  "code": 20000,
  "flag": true,
  "message": "更新成功"
}
```

4.2代码实现

（ 1 ）修改src/api/gathering.js，增加方法定义

```
//更新
update(id,pojo) {
  return request({
    url: `/gathering/gathering/${id}`,
    method: 'put',
    data:pojo
  });
}
```

（ 2 ）修改src/views/table/gathering.vue的js脚本部分

增加属性id

```

data(){
    return {
        .....
        id: ''//当前编辑的ID
    }
}

```

修改handleEdit,增加

```

this.id = id;

```

```

handleEdit(id) {
    this.dialogFormVisible = true //打开dialog

    if (id !== '') {
        gatheringApi.findById(id).then(response => {
            this.pojo = response.data //回显数据
            this.id = id
        })
    } else {
        this.pojo = {}
    }
}

```

创建handleUpdate()方法

```

handleUpdate() {
    this.dialogFormVisible = false //关闭dialog
    gatheringApi.update(this.id, this.pojo).then(response => {
        if(response.flag){
            alert(response.message);
            this.fetchData();
        }
    })
}

```

创建方法saveOrUpdate

```

saveOrUpdate() {
  if (this.id !== '') {
    this.handleUpdate()
  } else {
    this.handleSave()
  }
}

```

(3) 修改保存按钮的点击事件

```
<el-button type="primary" @click="saveOrUpdate()">保存</el-button>
```

5 消息提示框

js原生的alert简直是丑爆了，我们可以使用elementUI提供了消息提示框 alert(response.message)可以替换为以下代码：

```

this.$message({
  message: response.message,
  type: response.flag ? "success" : "error"
});

```

\$message详见官方文档：<http://element-cn.eleme.io/#/zh-CN/component/message>

第五章-活动管理-删除

1 需求分析

在表格的操作列增加“删除”按钮，点击删除按钮弹出提示框，确定后执行删除并刷新表格。

id	活动名称	活动地址	开始时间	结束时间	操作
dl8VZ1N	人速对一低	西藏自治区 阿里地区 噶尔县	2011-09-13	2011-09-13	修改 删除

2准备工作

修改 EasyMock接口 URL: gathering/:id Method: delete

```

{
  "flag": true,
  "code": 20000,
  'message': "删除成功"
}

```

3 代码实现

(1) 修改src/api/gathering.js，增加方法定义


```
//根据id删除
deleteById(id){
  return request({
    url: `/gathering/${id}`,
    method: 'delete'
  });
}
```

(2) 修改src/views/table/gathering.vue的js脚本部分

增加方法

```
//5.删除
handleDelete(id) {
  this.$confirm("您要删除当前数据吗?", "提示", {
    confirmButtonText: "确定",
    cancelButtonText: "取消",
    type: "warning"
  }).then(() => {
    gatheringAPI.deleteById(id).then(response => {
      this.$message({
        message: response.message,
        type: response.flag ? "success" : "error"
      });

      if (response.flag) {
        this.fetchData();
      }
    });
  })
  .catch(() => {
    this.$message({
      type: "info",
      message: "已取消删除"
    });
  });
}
```

(3) 修改src/views/table/gathering.vue , 在操作列增加删除按钮

```
<el-button @click="handleDelete(scope.row.id)" type="text" size="small" >删除</el-button>
```

第六章-代码优化

- 我们看一下现在的API代码

注意这里的`
如果使用`
可以匹配`\${group_name}`
则无法用`\${group_name}`引入`

```
url: `${group_name}/${api_name}/${id}`,
```

```
import request from '@utils/request'
```

```
export default {
```

```
  //删除
```

```
  delete(id){
    return request({
      url: `/gathering/gathering/${id}`,
      method: 'delete'
    })
  },
```

```
  //更新
```

```
  update(id, pojo) {
    return request({
      url: `/gathering/gathering/${id}`,
      method: 'put',
      data: pojo
    });
  },
```

```
  //根据id查询活动
```

```
  findById(id) {
    return request({
      url: `/gathering/gathering/${id}`,
      method: 'get'
    });
  },
```

```
  //新增活动
```

```
  save(pojo) {
    return request({
      url: '/gathering/gathering',
      method: 'post',
      data: pojo
    })
  },
```

```
  //获得活动列表
```

```
  getList() {
    return request({
      url: '/gathering/gathering',
      method: 'get'
    })
  },
```

```
  //获得分页  /gathering/gathering/search/{page}/{size}
```

```

search(page, size, searchMap) {
  return request({
    url: '/gathering/gathering/search/{page}/{size}',
    method: 'post',
    data: searchMap
  })
}

}

```

- 这里的url地址都是一样的，如果以后地址发生了变化，需要逐个修改，不利于维护，所以我们这里把此字符串提取出来定义为常量，运用es6的模板字符串特性来进行拼接即可。修改后代码如下：

```

import request from '@utils/request' //引入request
const group_name='gathering'
const api_name='gathering'

export default{
  //获得活动列表
  getList(){
    return request({
      url: `/${group_name}/${api_name}`,
      method: 'get'
    })
  },
  //分页获得获得列表
  search(page,size,searchMap){
    return request({
      url: `/${group_name}/${api_name}/search/${page}/${size}`,
      method: 'post',
      data: searchMap
    })
  },
  //添加活动
  save(pojo){
    return request({
      url: `/${group_name}/${api_name}`,
      method: 'post',
      data: pojo
    })
  },
  //根据id查询
  findById(id){
    return request({
      url: `/${group_name}/${api_name}/${id}`,
      method: 'get'
    });
  },
  //更新

```

```
update(id,pojo){
    return request({
        url:`/${group_name}/${api_name}/${id}`,
        method:'put',
        data:pojo
    });
},
//根据id删除
deleteById(id){
    return request({
        url:`/${group_name}/${api_name}/${id}`,
        method:'delete'
    });
}

}
```