[vue+Ueditor集成 [前后端分离项目][图片、文件上传][富文本编辑]](http://www.cnblogs.com/ocean-sky/p/7132319.html)

**写在最前面的话：鉴于近期很多的博友讨论，说我按照文章的一步一步来，弄好之后，怎么会提示后端配置项http错误，文件上传会提示上传错误。这里提别申明一点，ueditor在前端配置好后，需要与后端部分配合进行，后端部分的项目代码git地址：https://github.com/coderliguoqing/UeditorSpringboot，然后将配置ueditor.config.js 里的serverUrl的前缀改陈你自己的后端访问的请求路径地址，文件上传的后端部分，只提供了demo，具体对接文件服务器的部分需要自己修改完成。**

首先，谈下这篇文章中的前后端所涉及到的技术框架内容。

        虽然是后端的管理项目，但整体项目，是采用前后端分离的方式完成，这样做的目的也是产品化的需求；

        **前端，vue+vuex+vue router+webpack+elementUI的方案完成框架的搭建，其中用到了superUI来作为后端登陆之后的主页面框架，中间集成vue的大型单页应用；**

**后端，springboot+spring+springmvc+spring serurity+mybatis+maven+redis+dubbo**

**+zookeeper的方式来构建项目框架和管理，提供给前端restful风格的接口。此处还提供app端、PC WEB端的接口。**

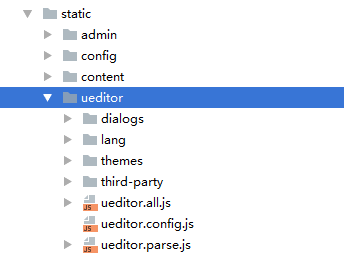
        UEditor之前一直有在项目中使用，作为国内开源的富文本编辑器，有百度的强大技术支持，整体来说是不错的选择，百度也提供了php、asp、.net、jsp的版本。原有的项目是采用整体式的开发方式，采用的是jsp的页面开发技术，所以集成起来相对来说更加容易，只需要按照文档的方式将前端集成进去，然后后端拿到源码之后，针对文件上传的类修改最终存储的方法即可将文件等上传到本身的服务器了。

     然而，由于决定了做前后端分离的方式，必然就会有新的坑，特别是还选择了新的技术vue.js+elementUI的这种方式。那么也只能放手一搏，不多啰嗦，介绍完，马上开始正事。

        **1、下载UEditor官网最新的jsp版本的包，下载完成解压之后得到一个ueditor1\_4\_3\_3-utf8-jsp的文件夹，里面包含的内容如下：**



**除了jsp的文件夹之外，其余的文件和文件夹复制到前端项目中的static用于存放静态文件的目录下，结构如下：**



这里特别说明jsp目录下的资源为何不放进来，因为我们是vue搭建的项目，jsp页面肯定是不会放在前端的项目中的，包括config.json也放在后端用于解析，这里后面会解释这样做的原因。

     **2、前端将文件放进来之后，暂时先这样，咱们来整理后端的东西。这里将jsp目录下的lib目中的ueditor.jar文件中的所有类全部拿出来（具体方式自己决定，反编译工具或者拿到源码都可以）,放到后端项目中，然后在control层新建一个UeditorController.java的类，如下：**

1 /\*\*
2 \* 用于处理关于ueditor插件相关的请求
3 \* @author Guoqing
4 \*
5 \*/
@RestController

@CrossOrigin

@RequestMapping("/sys/ueditor")

public class UeditorController extends BaseController {

@RequestMapping(value = "/exec")

@ResponseBody

public String exec(HttpServletRequest request) throws UnsupportedEncodingException{
request.setCharacterEncoding("utf-8"); String rootPath = request.getRealPath("/");

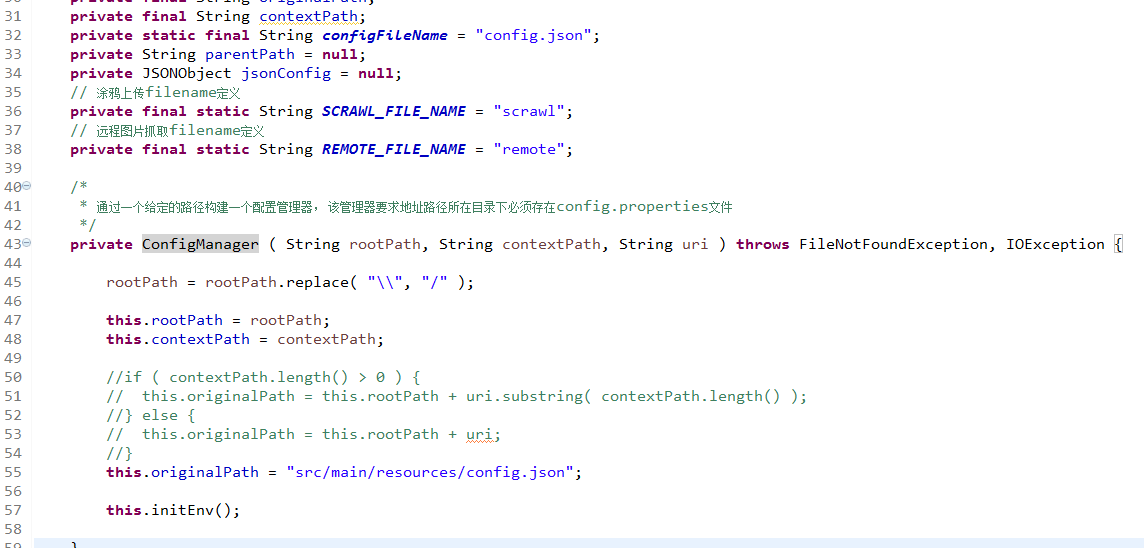
return new ActionEnter( request, rootPath).exec();

}
}

该类主要处理，ueditor与后端服务器的交互，通过action=''不同的类型来处理，其中action=config为加载配置项，action=uploadImg 图片上传，在ActionEntor类中，你可以根据不同的请求类型来处理；

处理当action=config时的情况，保证前端的编辑器各项文件上传功能，能够正常使用。

然后jsp目录下的config.json文件放到java/main/resources目录下，修改ConfigManager.java类，如下：



注释掉原有的读取配置文件的方式，添加新的读取路径，这样确保ueditor在初始化能够正确的加载配置文件。此时，修改前端项目中ueditor.config.js中的serverUrl的值为:

// 服务器统一请求接口路径
, serverUrl: "http://localhost:8080/sys/ueditor/exec"

而，针对ActionEnter.java类中，如下代码后的文件上传的处理，请大家针对自身的上传方式和文件服务器选择适合自己的方式：

switch ( actionCode ) {
//读取配置文件时的请求处理
case ActionMap.CONFIG:
return this.configManager.getAllConfig().toString();
//上传图片、视频、文件时的处理
case ActionMap.UPLOAD\_IMAGE:
case ActionMap.UPLOAD\_SCRAWL:
case ActionMap.UPLOAD\_VIDEO:
case ActionMap.UPLOAD\_FILE:
conf = this.configManager.getConfig( actionCode );
state = new Uploader( request, conf, baseFileService ).doExec();
break;
//抓取远程图片时的处理方式，此处也可以关闭
//当从网页上复制内容到编辑器中，如果图片与该域名不同源，则会自动抓到本地的服务器保存
case ActionMap.CATCH\_IMAGE:
conf = configManager.getConfig( actionCode );
String[] list = this.request.getParameterValues( (String)conf.get( "fieldName" ) );
state = new ImageHunter( conf ).capture( list );
break;
//上传多文件时的文件在线管理
case ActionMap.LIST\_IMAGE:
case ActionMap.LIST\_FILE:
conf = configManager.getConfig( actionCode );
int start = this.getStartIndex();
state = new FileManager( conf ).listFile( start );
break;
}
return state.toJSONString();

接下来是前端的处理，介于大家的要求。作者将原本的demo进行了优化，将编辑器封装成组件的方式，方便调用，代码如下：

<template>
<div>
<script id="editor" type="text/plain" ></script>
</div>
</template>
<script>
import AppConfig from '@/config'
import '../../../../../../static/ueditor/ueditor.config.js'
import '../../../../../../static/ueditor/ueditor.all.js'
import '../../../../../../static/ueditor/lang/zh-cn/zh-cn.js'
export default {
name: "UEditor",
props: {
id: {
type: String
},
config: {
type: Object
}
},
data() {
return {
editor: null
}
},
mounted() {
//初始化UE
const \_this = this;
this.editor = UE.getEditor('editor',this.config);
},
destoryed() {
this.editor.destory();
},
methods:{
getUEContent: function(){
return this.editor.getContent();
}
}
}
</script>

 导出组件：

var UEditor = require('./src/ueditor.vue');
module.exports = {
UEditor
}

这里之所以是用一个模态框的方式来加载编辑器，是因为会存在编辑器工具栏浮动的问题，如果有问题，请根据如下的配置来处理即可；

页面调用：

<template>
<div id="app" class="hello">
<el-button size="primary" type="info" icon="plus" @click="openWindow">打开窗口</el-button>
<el-dialog title="新增菜单" size="small" v-model="addFormVisible" :close-on-click-modal="false">
<div>
<el-button size="primary" type="info" icon="plus" @click="getContent">获取内容</el-button>
<UEditor :config=config ref="ueditor"></UEditor>
</div>
</el-dialog>
</div>
</template>
<script>
import {UEditor} from './ueditor/index.js'
export default{
name: 'hello',
components: {UEditor},
data(){
return {
config: {
/\*//可以在此处定义工具栏的内容
toolbars: [
['fullscreen', 'source','|', 'undo', 'redo','|','bold', 'italic', 'underline', 'fontborder', 'strikethrough',
'|','superscript','subscript','|', 'forecolor', 'backcolor','|', 'removeformat','|', 'insertorderedlist', 'insertunorderedlist',
'|','selectall', 'cleardoc','fontfamily','fontsize','justifyleft','justifyright','justifycenter','justifyjustify','|',
'link','unlink']
],\*/
autoHeightEnabled: false,
autoFloatEnabled: true,　　//是否工具栏可浮动
initialContent:'请输入内容', //初始化编辑器的内容,也可以通过textarea/script给值，看官网例子
autoClearinitialContent:true, //是否自动清除编辑器初始内容，注意：如果focus属性设置为true,这个也为真，那么编辑器一上来就会触发导致初始化的内容看不到了
initialFrameWidth: null,
initialFrameHeight: 450,
BaseUrl: '',
UEDITOR\_HOME\_URL: 'static/ueditor/'
},
addFormVisible: false
}
},
methods: {
openWindow: function(){
this.addFormVisible = true;
},
//获取文档内容
getContent: function(){
let content = this.$refs.ueditor.getUEContent();
console.log(content);
alert(content);
}
}
}
</script>

至此，大功告成，包括文件上传下载等部分全部搞定，不过要声明一点的是，当出现接口与页面部署域名不同时，点击选择图片上传会出现iframe跨域的问题。

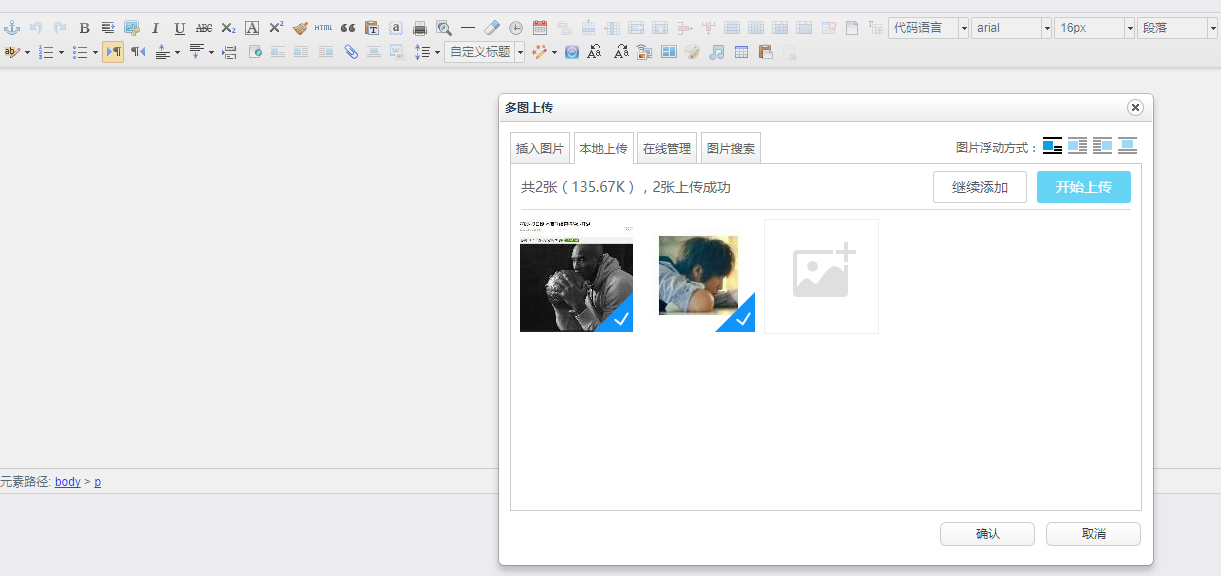
2017-09-08的更新，作者已经通过更改源码的方式，处理了单图选择文件上传存在跨域的问题，处理方式当然就是将原有的form.submit的表单上传方式，更改为ajax的上传方式；

ueditor.all.js 24503行的方法替换，源码如下：

　　　　　 /\*\*
\* 2017-09-07 改掉了ueditor源码，将本身的单文件上传的方法改为ajax上传，主要目的是为了解决跨域的问题
\* @author Guoqing
\*/
domUtils.on(input, 'change', function() {
if(!input.value) return;
var loadingId = 'loading\_' + (+new Date()).toString(36);
var imageActionUrl = me.getActionUrl(me.getOpt('imageActionName'));
var allowFiles = me.getOpt('imageAllowFiles');
me.focus();
me.execCommand('inserthtml', '<img class="loadingclass" id="' + loadingId + '" src="' + me.options.themePath + me.options.theme +'/images/spacer.gif" title="' + (me.getLang('simpleupload.loading') || '') + '" >');
/!\* 判断后端配置是否没有加载成功 \*!/
if (!me.getOpt('imageActionName')) {
errorHandler(me.getLang('autoupload.errorLoadConfig'));
return;
}
// 判断文件格式是否错误
var filename = input.value,
fileext = filename ? filename.substr(filename.lastIndexOf('.')):'';
if (!fileext || (allowFiles && (allowFiles.join('') + '.').indexOf(fileext.toLowerCase() + '.') == -1)) {
showErrorLoader(me.getLang('simpleupload.exceedTypeError'));
return;
}
var params = utils.serializeParam(me.queryCommandValue('serverparam')) || '';
var action = utils.formatUrl(imageActionUrl + (imageActionUrl.indexOf('?') == -1 ? '?' : '&') + params);
var formData = new FormData();
formData.append("upfile", form[0].files[0] );
$.ajax({
url: action,
type: 'POST',
cache: false,
data: formData,
processData: false,
contentType: false,
success: function (data) {
data = JSON.parse(data);
var link, loader,
body = (iframe.contentDocument || iframe.contentWindow.document).body,
result = body.innerText || body.textContent || '';
link = me.options.imageUrlPrefix + data.url;
if(data.state == 'SUCCESS' && data.url) {
loader = me.document.getElementById(loadingId);
loader.setAttribute('src', link);
loader.setAttribute('\_src', link);
loader.setAttribute('title', data.title || '');
loader.setAttribute('alt', data.original || '');
loader.removeAttribute('id');
domUtils.removeClasses(loader, 'loadingclass');
} else {
showErrorLoader && showErrorLoader(data.state);
}
form.reset();
}
});
function showErrorLoader(title){
if(loadingId) {
var loader = me.document.getElementById(loadingId);
loader && domUtils.remove(loader);
me.fireEvent('showmessage', {
'id': loadingId,
'content': title,
'type': 'error',
'timeout': 4000
});
}
}
});

好了，如果以上对你有帮助的话，请顺手点个赞，谢谢各位大虾们啦！

效果图如下：





 各位如果还有问题的，那么请入群吧： **qq   581036486**

https://github.com/coderliguoqing