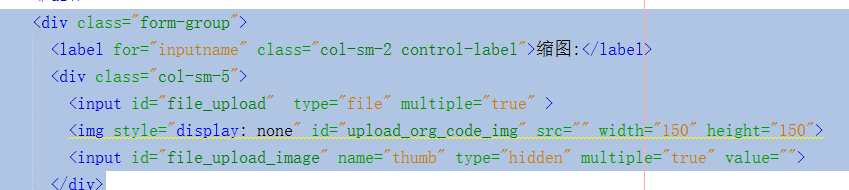
# 图片上传

图片的上传是通过image.js触发的 当用户点击上传图片以后会触发如下：

具体的html代码;

/\*\*

\* 图片上传功能

\*/

$(function() {

$('#file\_upload ').uploadify({

//上传图片以后的动画 定义了动画地址 在点击的页面SCOPE的域下定义

**'swf' : SCOPE.ajax\_upload\_swf,**

//js图片上传请求地址

'uploader' : SCOPE.ajax\_upload\_image\_url,

'buttonText': '图片上传',

'fileTypeDesc': 'Image Files',

//文件名称 model层的时候需要

// return '/' .self::UPLOAD . '/' . $res['imgFile']['savepath'] . $res['imgFile']['savename'];

'fileObjName' : 'file',

//允许上传的文件后缀

'fileTypeExts': '\*.gif; \*.jpg; \*.png',

//文件上传以后 会调用此

'onUploadSuccess' : function(file,data,response) {

// response true ,false

if(response) {

var obj = JSON.parse(data); //由JSON字符串转换为JSON对象

console.log(data);

//上传完成以后显示的信息

$('#' + file.id).find('.data').html(' 上传完毕');

//获取到图片的路径节点 将data数据插入

$("#upload\_org\_code\_img").attr("src",obj.data);

//隐藏域传递图片的信息

$("#file\_upload\_image").attr('value',obj.data);

//图片的展示 默认隐藏

$("#upload\_org\_code\_img").show();

}else{

alert('上传失败');

}

},

});

});

当上传以后 会触发

function ajaxuploadimage() {

//实例化该方法

$res = D('UploadImage')->imageUpload();

if ($res == FALSE) {

return show(0, "上传失败！");

}

return show(1,"上传成功了！",$res);

}

Model层设计

public function imageUpload() {

$res = $this->\_uploadObj->upload();

//如果上传成功 返回文件详细的路径

if($res) {

return '/' .self::UPLOAD . '/' . $res['file']['savepath'] . $res['file']['savename'];

}else{

return false;

}

}

Js获取html的页面中的上传图片的ID，进行js操作

/\*\*

\* 图片上传功能

\*/

$(function() {

$('#file\_upload').uploadify({

'swf' : SCOPE.ajax\_upload\_swf,

'uploader' : SCOPE.ajax\_upload\_image\_url,

'buttonText': '上传图片',

'fileTypeDesc': 'Image Files',

'fileObjName' : 'file',

//允许上传的文件后缀

'fileTypeExts': '\*.gif; \*.jpg; \*.png',

'onUploadSuccess' : function(file,data,response) {

// response true ,false

if(response) {

var obj = JSON.parse(data); //由JSON字符串转换为JSON对象

console.log(data);

$('#' + file.id).find('.data').html(' 上传完毕');

$("#upload\_org\_code\_img").attr("src",obj.data);

$("#file\_upload\_image").attr('value',obj.data);

$("#upload\_org\_code\_img").show();

}else{

alert('上传失败');

}

},

});

});

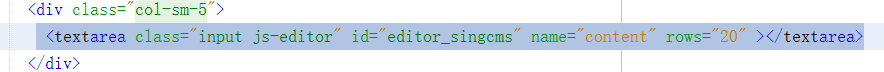
去请求add.html下边定义的uploader

# 编辑器内部图片的上传

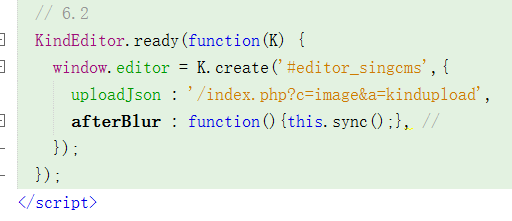
首先需要加载编辑器的插件



定义一个ID



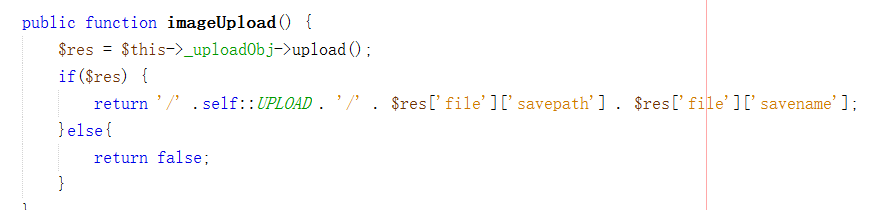
与下边的ID是对应的



当用户上传图片以后，就会去触发该接口

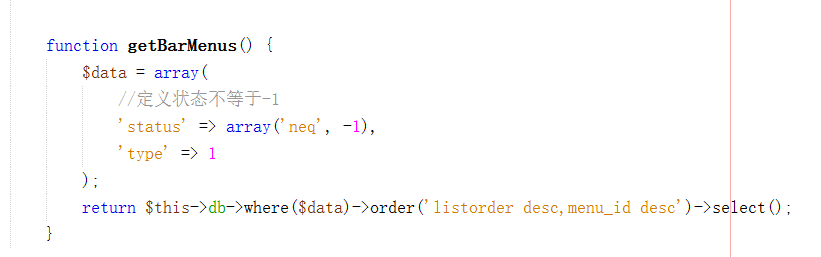


调用model层 自身的图片上传函数



文章管理 数据传递到模板

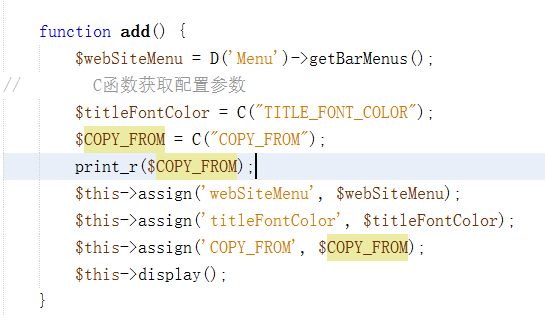
首先查询栏目数据 需要status不为1 type为1



在控制器层 实例化对象 将数据传递到模板

要把数据传递到哪里 就在那也页面展示的方法下 进行查询数据

Add页面需要展示数据 就需要在add的方法下查询



数据的填充



定义foreach一定需要定义item属性



# 文章管理Add页面提交数据到服务器

当点击提交的时候 由于在common.js中定义了触发如下



此时会向该页面的save\_url的地址 进行数据提交