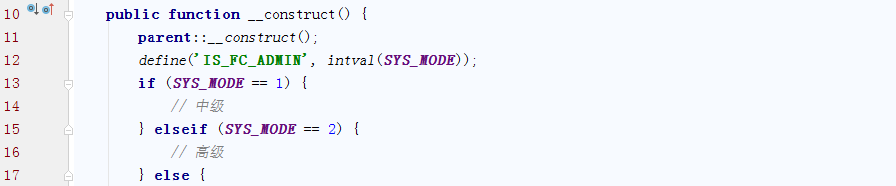
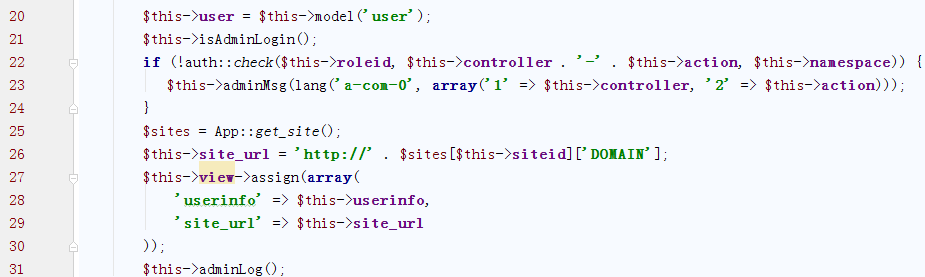
Admin

\_\_construct



由此可见，选择级别并没有用

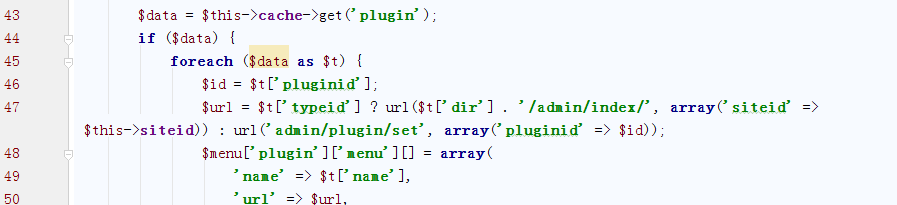


检查权限，获取站点信息，记录日志

sysMenu



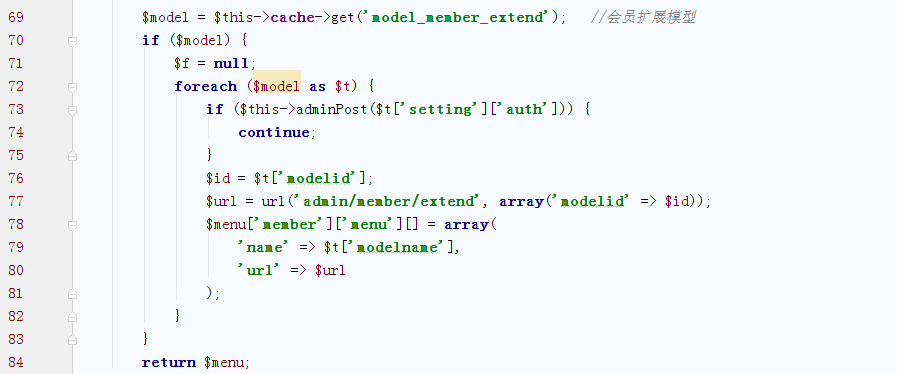
获取右侧栏目配置，如果设置站点关闭，销毁



获取插件缓存

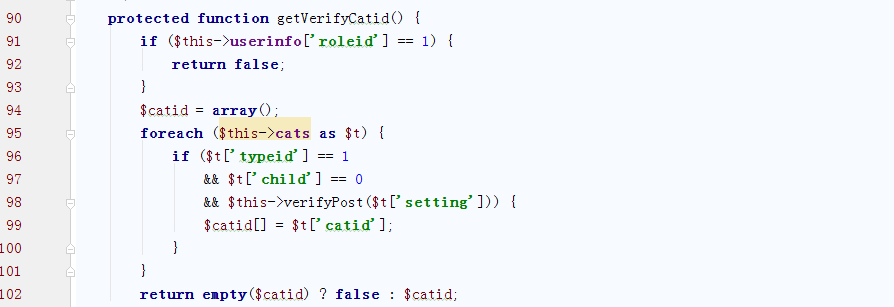


获取表单模型，检查权限，设置menu



获取用户扩展模型，组成会员menu

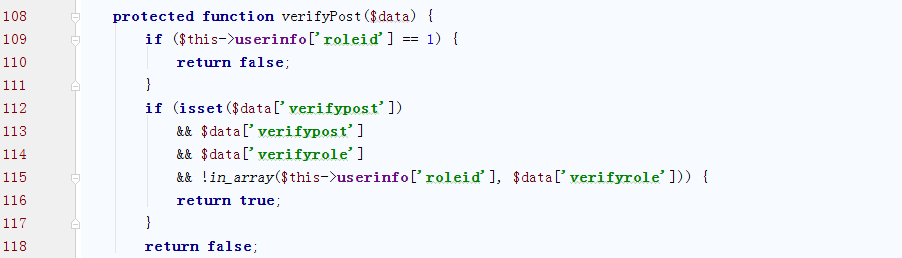
getVerifyCatid



角色id

遍历栏目，返回是内容栏目、且没有子栏目、且用户有投稿权限的catid

verifyPost



如果设置有投稿权限，且用户投稿权限在设置中

adminPost



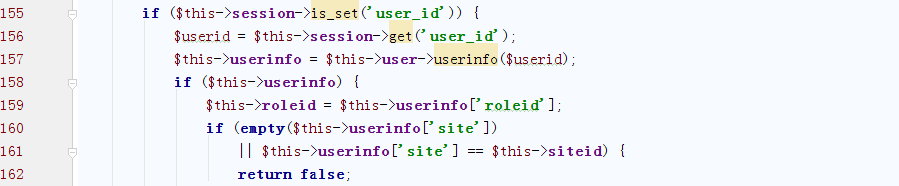
设置有后台投稿权限，用户有该权限；或设置有siteuser，且当前站点为配置站点

isAdminLogin

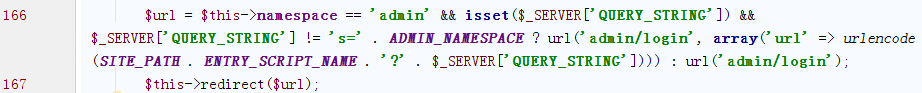
【只是检查是否在admin目录下，不检查权限。设置权限？159行】



检查目录、控制器、是否登陆



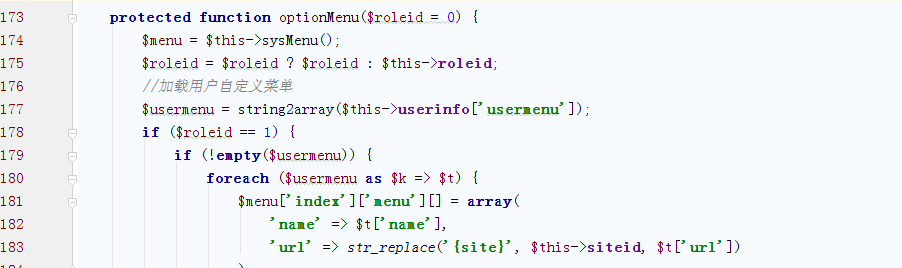
获取user\_id及用户信息，获取角色，检查用户的site是否为当前siteid，是则返回false



如果不为admin目录，跳转到登陆

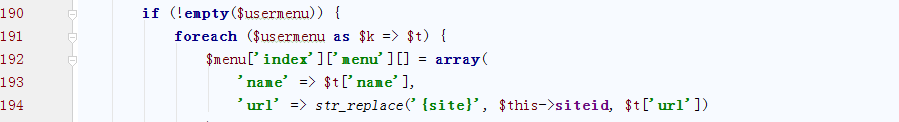
（显然，返回false继续进行，没有返回则跳转到login）

optionMenu

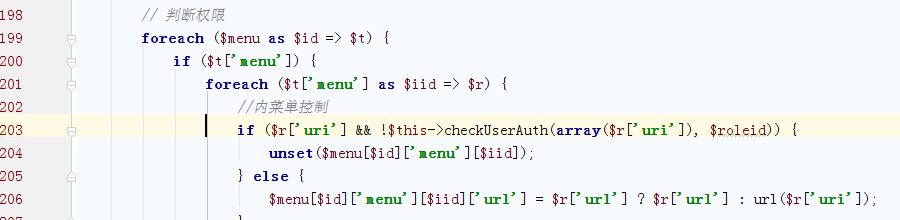




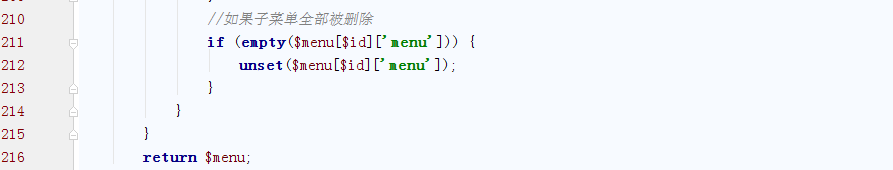
用户用户角色、menu，如果为1（全部权限）？重新组装menu



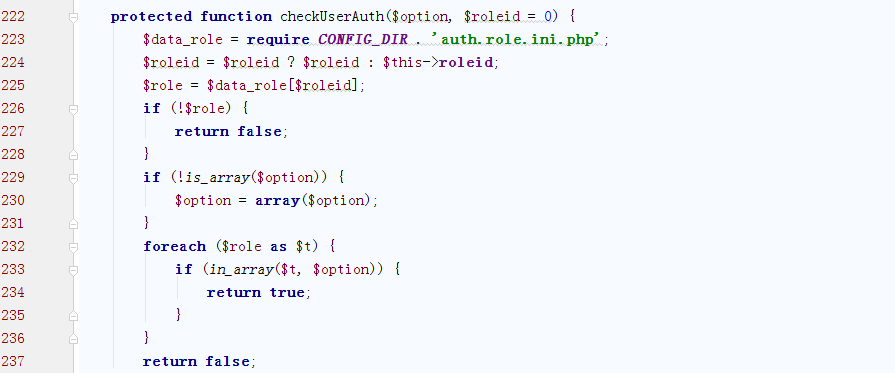
如果用户menu不为空，遍历，组装（有区别吗？区别在于下面过滤了没有权限的菜单）



遍历，检查用户权限，删掉没有权限的menu，或生成url



checkUserAuth

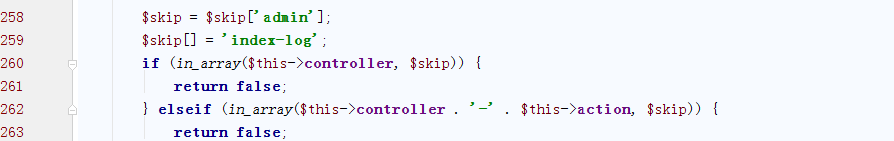


获取用户角色配置，检查权限

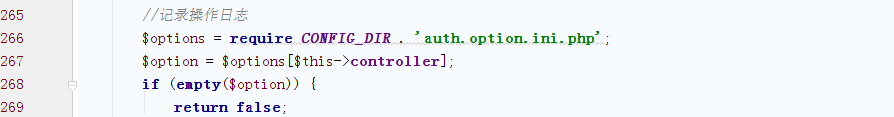
adminLog



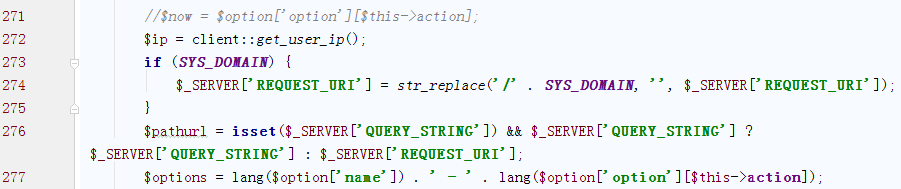
过滤不需要记录的情况



过滤不需要记录情况



获取每个控制器的权限清单，检查当前控制器是否在其中



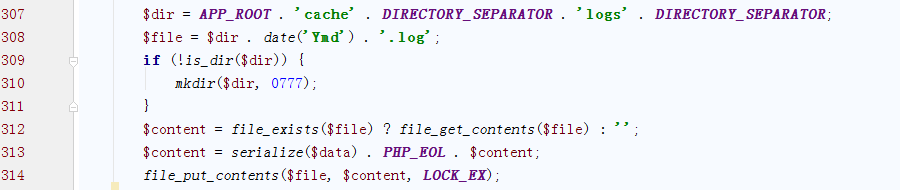
获取ip，uri，query，option



继续拼接提交转台option



组装数据

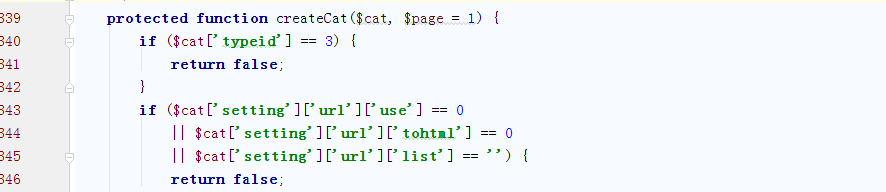


写入缓存日志

delDir

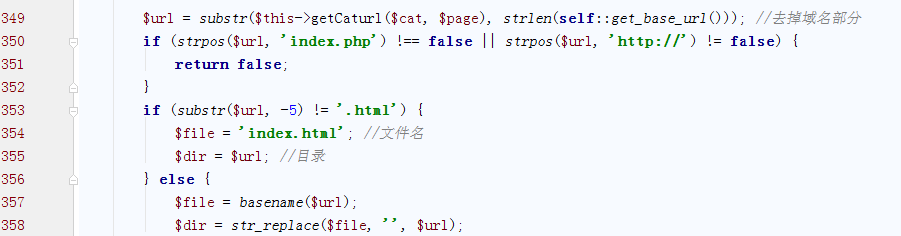
createCat

【按页生成缓存文件，前提是开启生成html静态文件配置，分页如何实现没看明白】

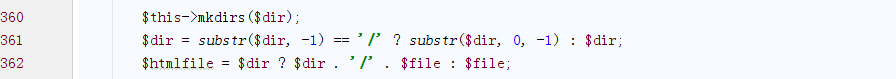


过滤外链

如果没有开启用户自定义url，或没有开启生成静态页面，或没有设置list，退出



获取url，设置文件名和目录，url以.html结尾



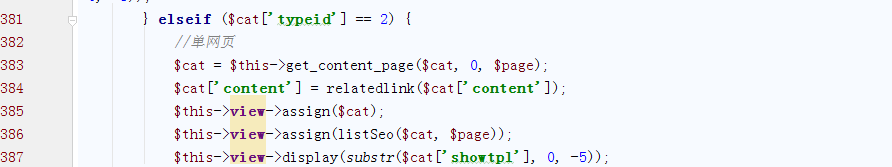
创建目录，文件名



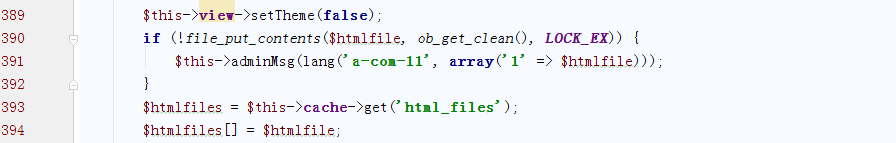
**开启缓存**

设置风格，获取cat缓存

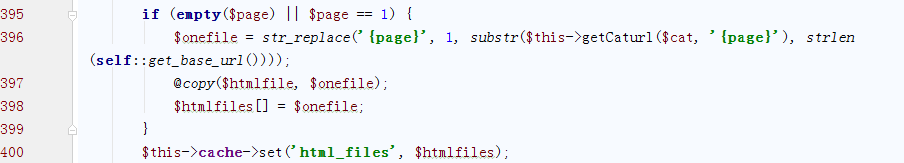
如果为内容页



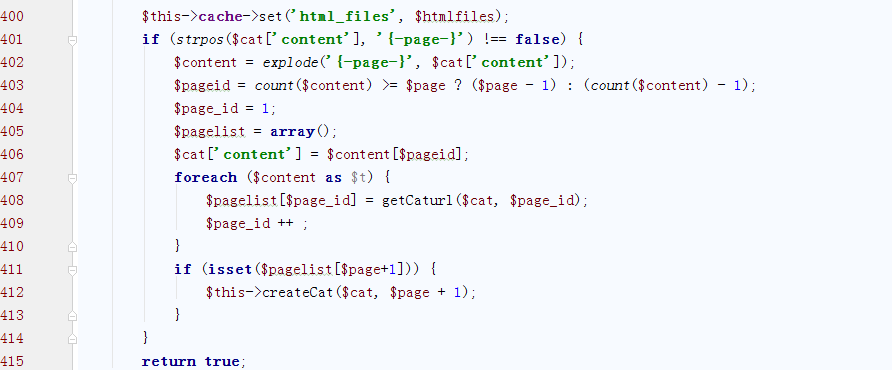
如果为单页，将cat传入get\_content\_page进行分页处理



写入html文件，获取文件



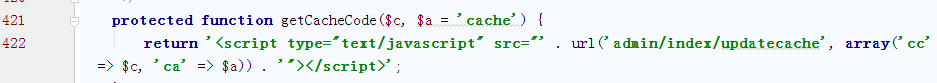
如果为没有页码或为第1页，生成第1页，复制文件，保存到缓存中



按页面分割，查看多少页，组装栏目

如果有下一页，递归调用本方法生成下一页

getCacheCode

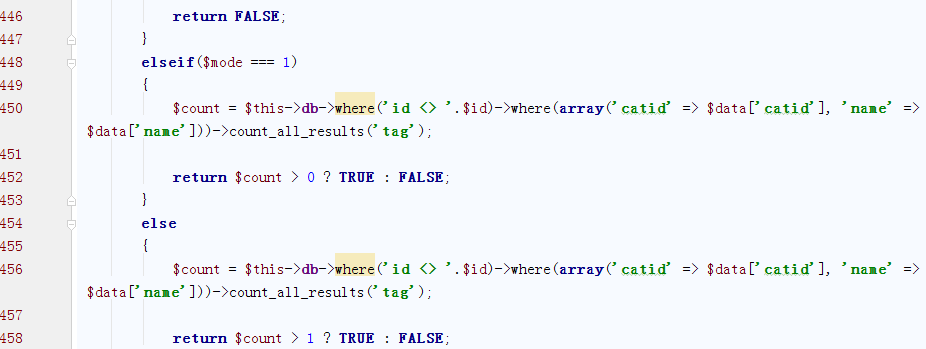


获取更新缓存JS代码

checkRepeat



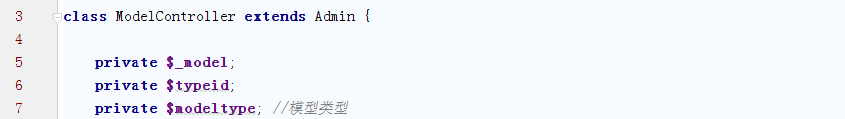
获取$data中的1个元素，然后销毁$data中的这个元素，然后再检查这个元素$data中是否还存在



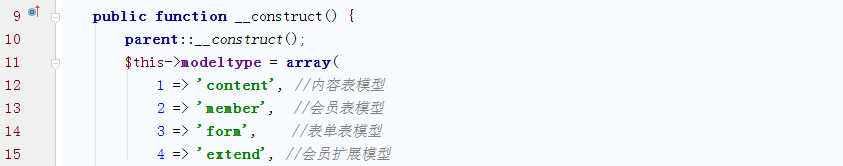
通过检查tag表是否存在id不相同，但catid和name相同的情况

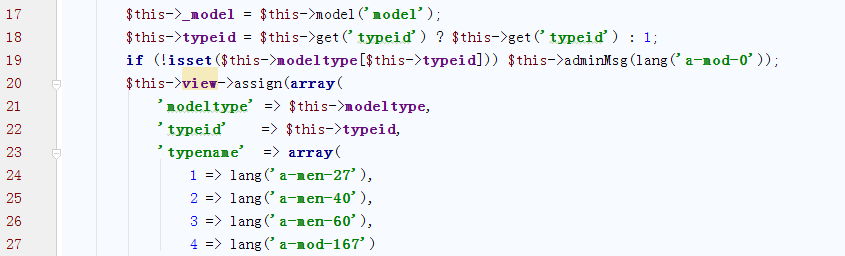
IndexController

ModelController



\_\_construct





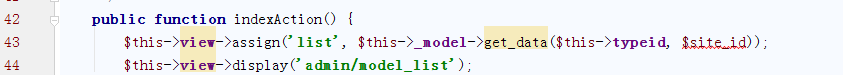
默认模型类型为1即内容模型

get\_model\_site\_id



有继承站点则为继承站点id，否则为自己的id

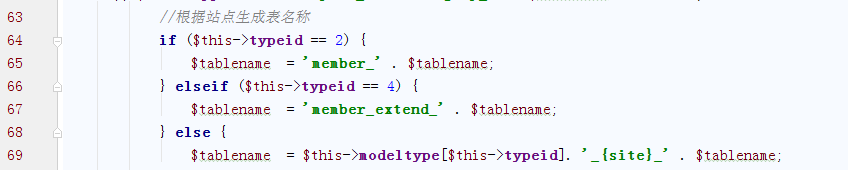
indexAction



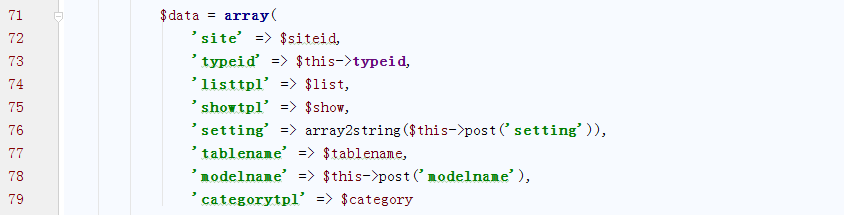
addAction

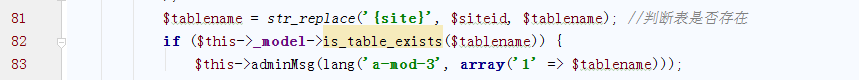


获取设置的表名（表名应该以数字结尾），列表模板、内容模板、栏目模板（如果为表单或会员扩展模型，默认为post\_模板，否则为category\_模板）



生成表名，包含模型类型、站点占位符（用于替换），其中会员和会员扩展模型没有站点名称





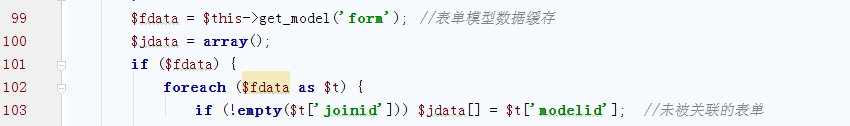
替换表名，检查表是否已存在



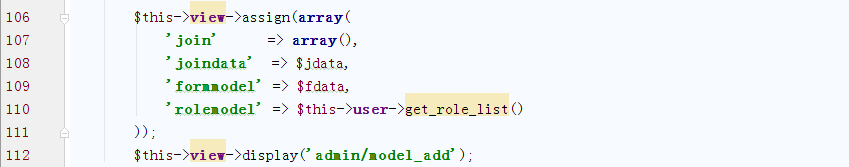
生成表

如果为内容模型，检查是否有入参关联模型，如果有，**更新**对应的关联模型（将model表中id=被关联的$join的那个字段的joinid设为新添加的这个model的id）

【不明白为什么setting输入只有2个值，最后显示完了就有所有的值了】

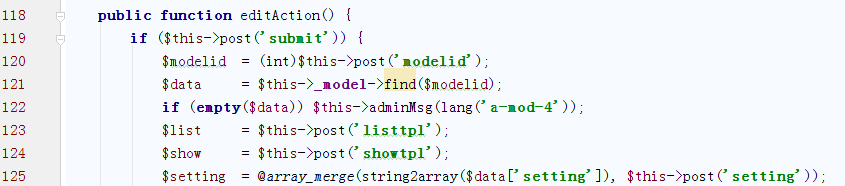


获取表单模型缓存，获取已关联的表单模型id用于显示



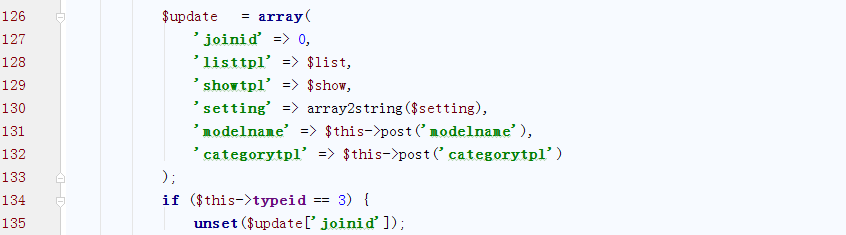
【总结】新增模型，就是（1）将填写的字段新增到model表中；（2）创建对应的表；（3）如果有站点继承本站点，创建对应站点的表；（4）如果新建的模型为内容模型，且关联了其他模型，设置对方模型的joinid为新建模型的modelid（显然，被关联的表必须还是表单模型，要不然根本就不会显示）

editAction

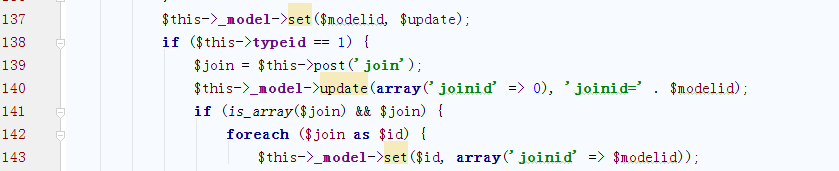


检查编辑的模型是否存在

获取修改的列表、内容、设置信息（设置信息和之前的合并）



如果为表单模型，销毁关联

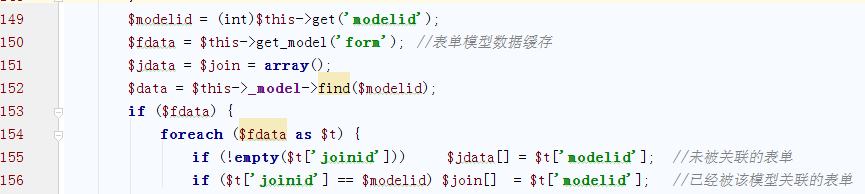


更新数据库

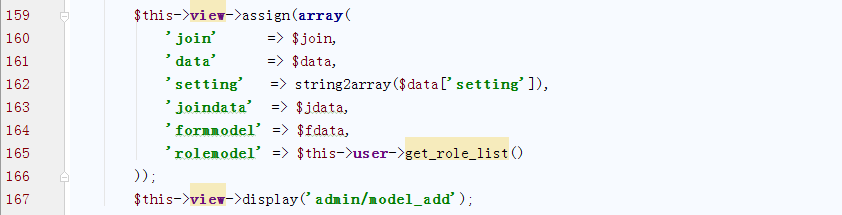
如果为内容模型，获取关联表单，先将joinid=$modelid的记录的joinid全置为0，然后在将获取的关联的表单id对应的记录的joinid全部置为$modelid



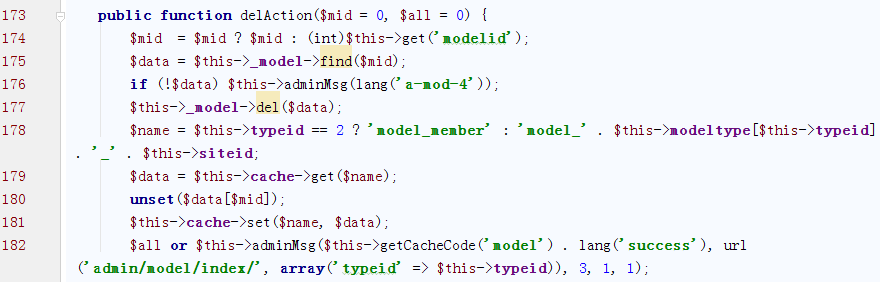
（以上是修改post情况）



获取模型数据，表单数据（关联和未被关联的，为空才是未被关联的？）



delAction



首先检查模型是否存在

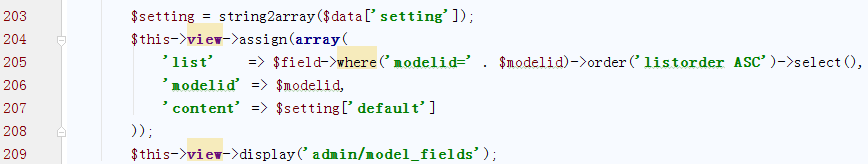
删除model表中模型数据，如果为会员模型，删除对应的缓存信息（都在model的del方法中删除了）

fieldsAction

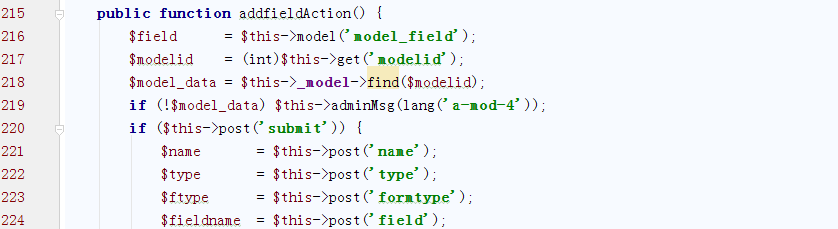
对表的字段进行排序



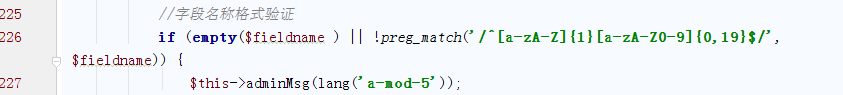
首先检查模型是否存在，获取对应表的模型、字段表模型，遍历修改字段，修改顺序



addfieldAction



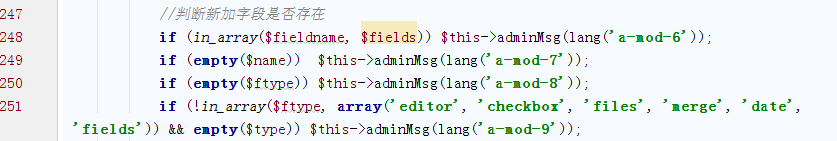
获取模型字段表、模型表数据，检查是否存在，获取修改的字段信息



检查字段格式是否采用字母+数字形式



获取主表、附表字段（两者合并作为最后添加的字段）

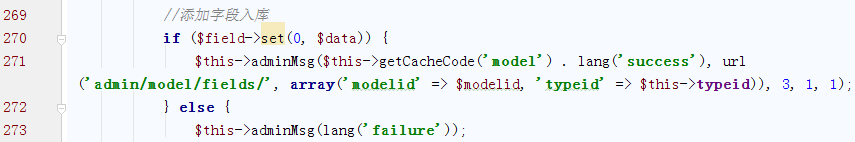


判断字段是否存在等



设置数据（均为model\_field表中的字段）

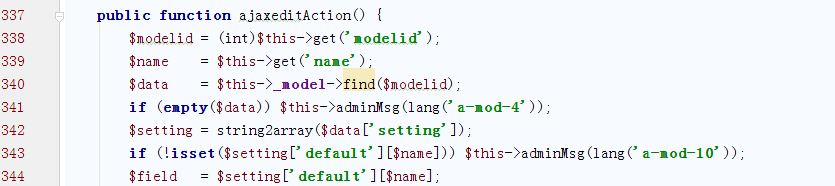
【注意】setting最后在页面中显示的就是“相关参数”，就表单项设置



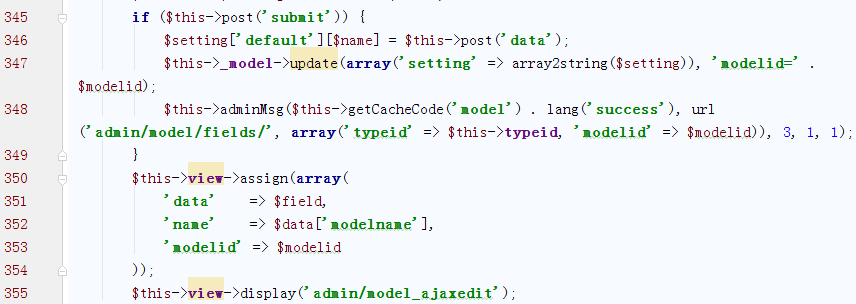
将数据插入model\_field表中（set中也同时修改了附表中的字段）



ajaxeditAction

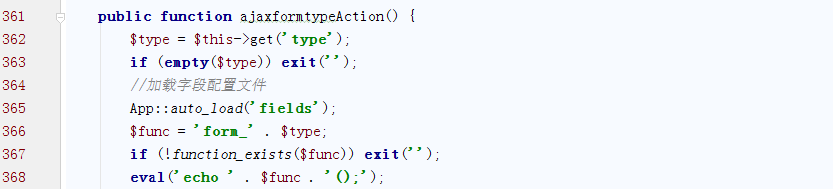


检查模型setting中的default是否有该字段，获取对应的name

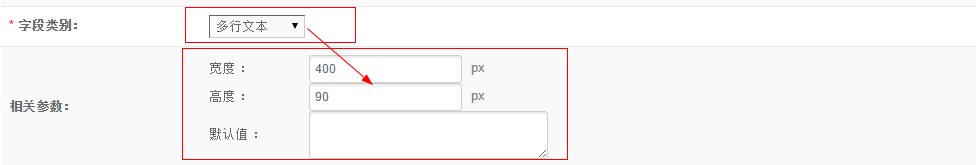


如果为提交，重新设置配置

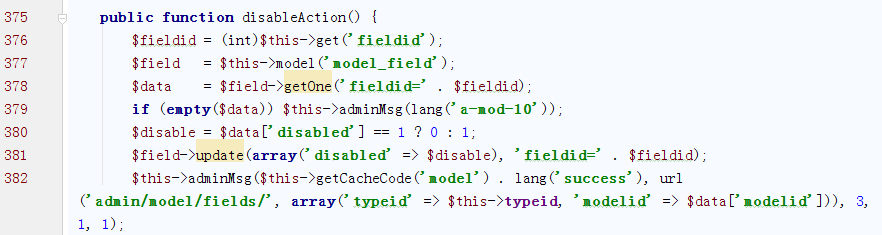
ajaxformtypeAction



根据字段类型的选择，获取对应的表单类型返回

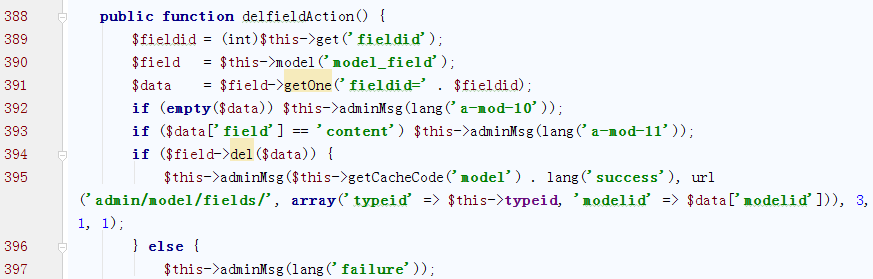


disableAction



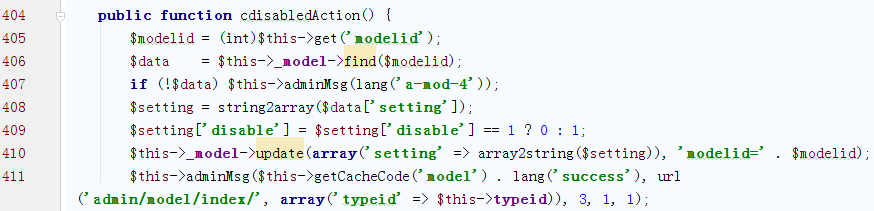
将字段值设为disabled

delfieldAction



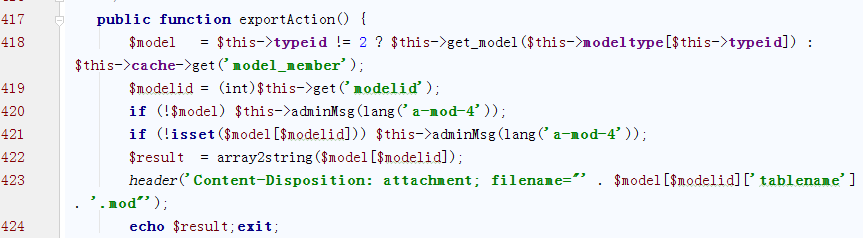
如果字段名为content，返回“字段不存在”

cdisabledAction



禁用/启用模型

exportAction



导出模型，如果为会员模型，直接获取缓存

importAction



检查表名、是否已存在



获取导入文件内容，转为数组



检查模型类型

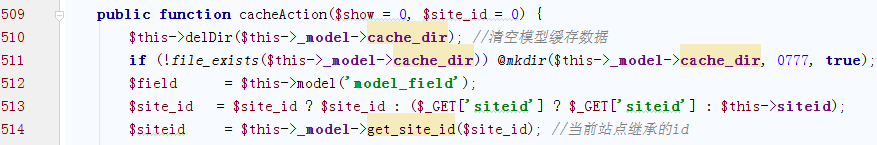


插入数据



更新数据

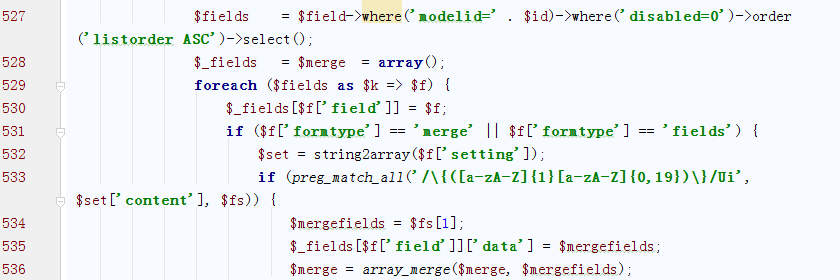
cacheAction



删除模型缓存，创建缓存文件夹，获取模型字段模型



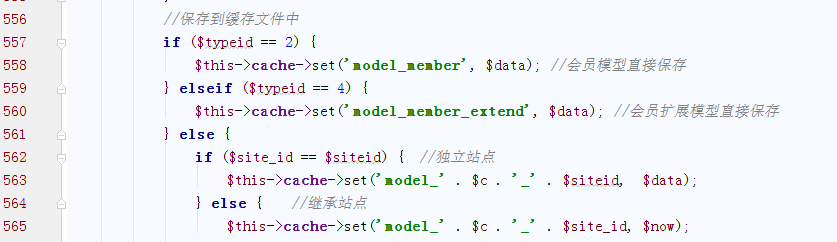
遍历模型类别，获取对应类别的模型数据，遍历，（1）首先获取设置，如果为disabled，continue；如果为继承站点，重新设置站点id和表名，整个缓存数组的id为model类型



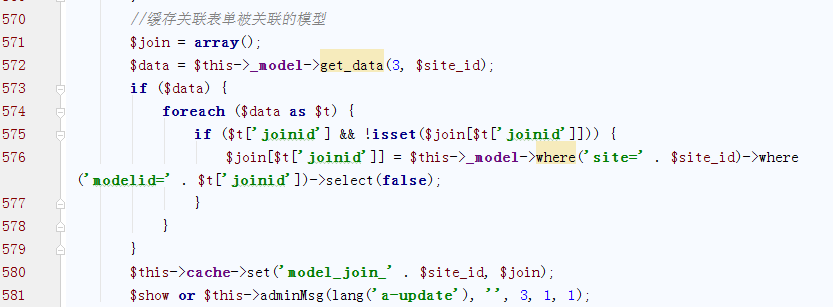
（2）设置fields，获取model\_field表中数据，遍历，如果表单类型为merge或fields，获取设置中的content，匹配字母和数字子组，设为[field][data]的值



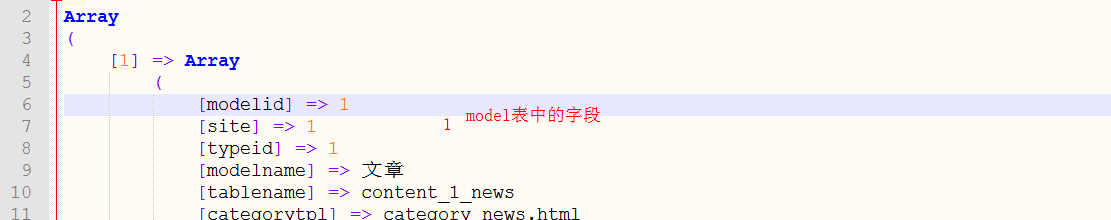
如果为内容模型，重新设置default，更新model表的setting字段

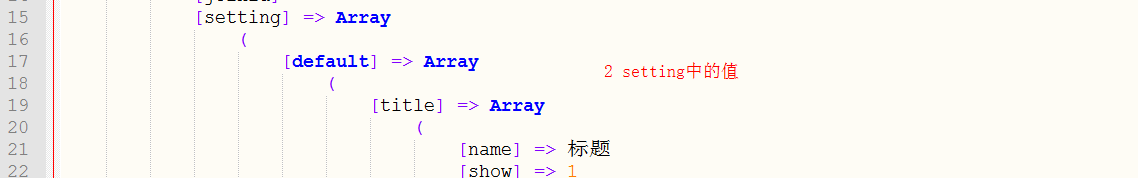


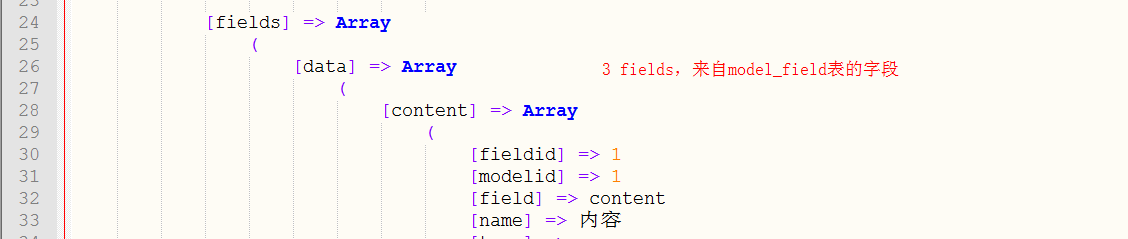
会员模型和会员扩展模型直接更新缓存，其他，加上站点id

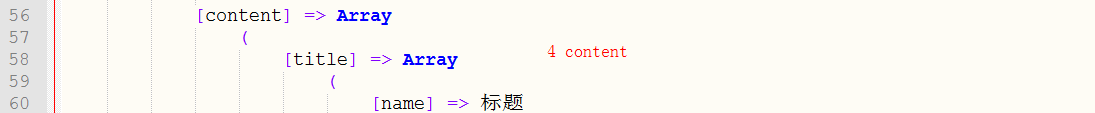


获取表单模型，遍历，如果设置有joinid，获取关联模型，保存缓存

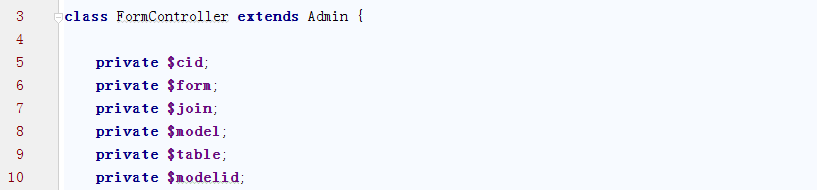




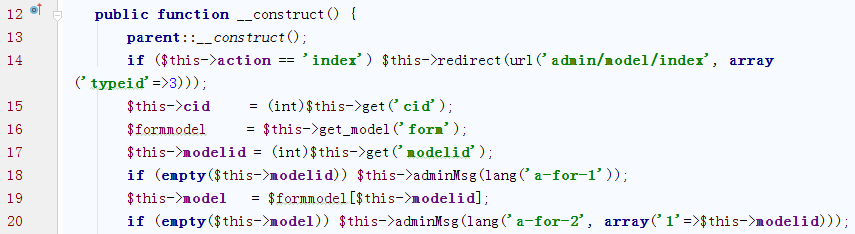




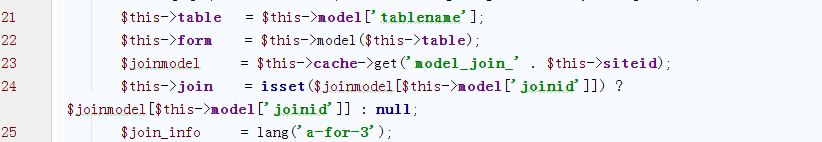
FormController



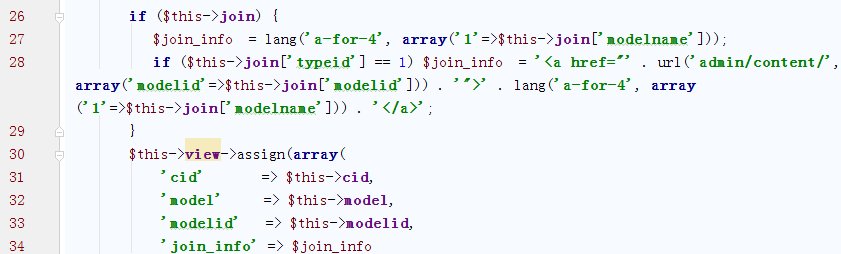
\_\_construct



如果a为index，跳转到model控制器



获取表名，对应的表model，关联



如果有关联，如果关联的是内容模型，设置a标签

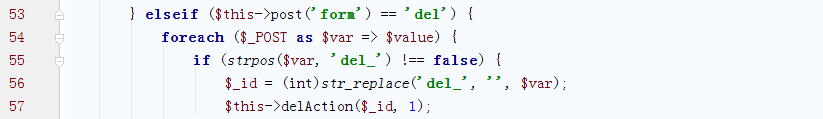
listAction

【获取关联文章】

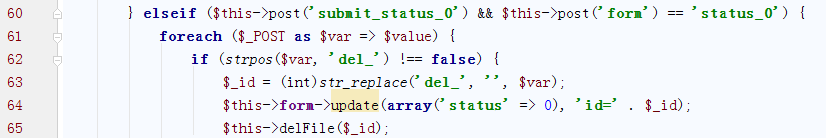


如果为搜索，获取关键字等

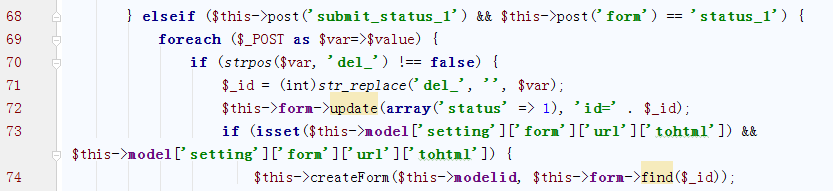
如果为排序，更新排序



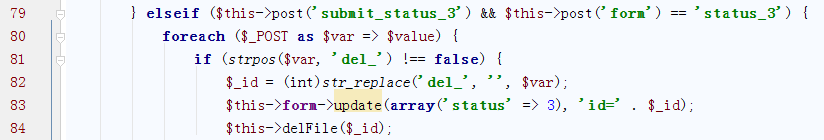
如果为删除



标记为未审



标记为审核



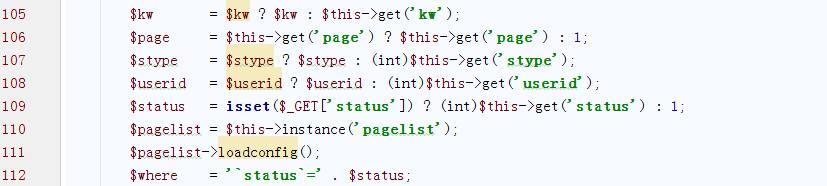
放入回收站



批量关联，批量关联应该是关联到对应的表模型。90行跟content表没关系，就是从join[tablename]中查找对应的关联模型。先更新form表的cid。

如果model模型的设置有生成url的html，生成表单html

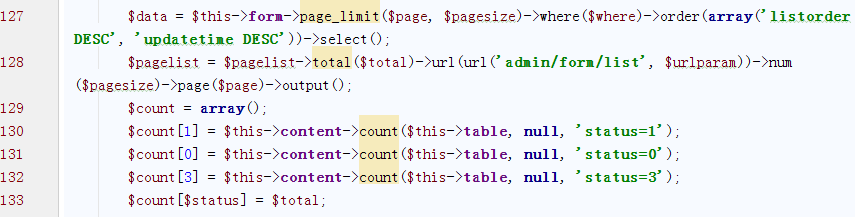




如果为搜索，获取搜索选项



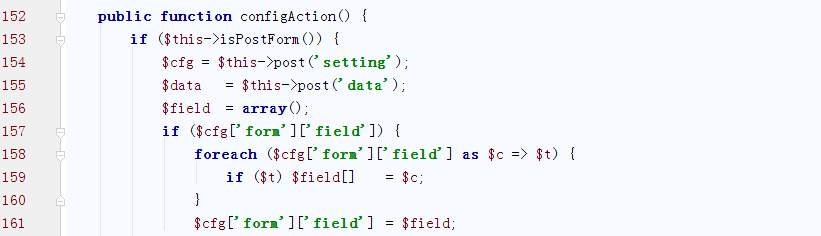
组合sql语句，查询内容表



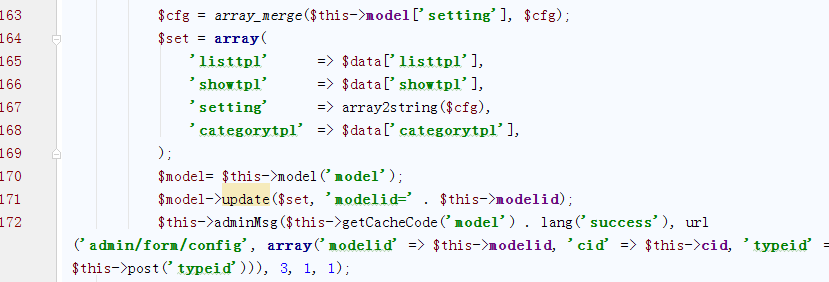
查询form模型，



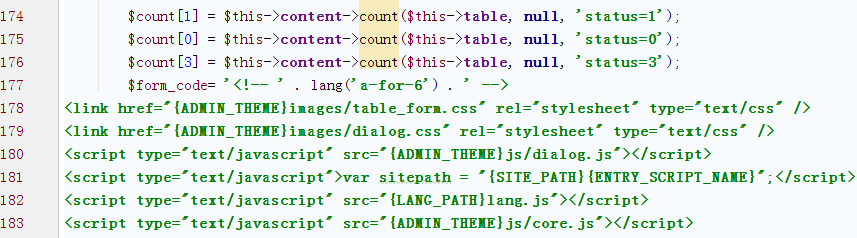
configAction



获取表单配置字段



合并原表单配置，更新到数据库





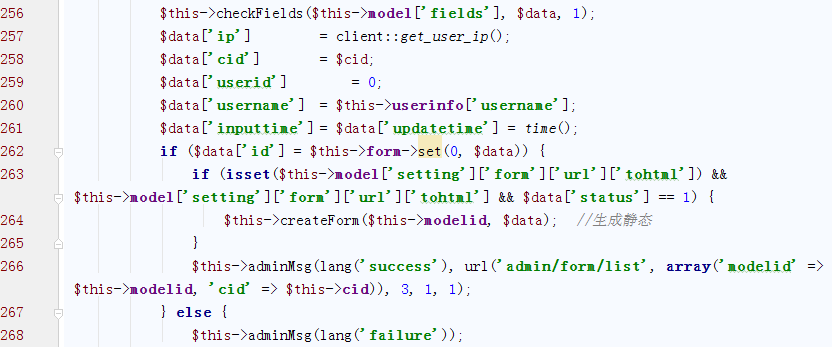
显示list工具

addAction

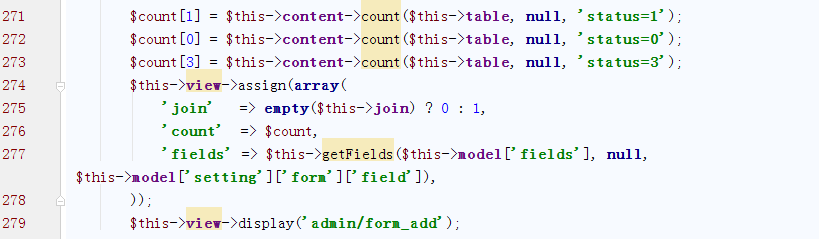


检查权限，获取数据，必须要关联才行

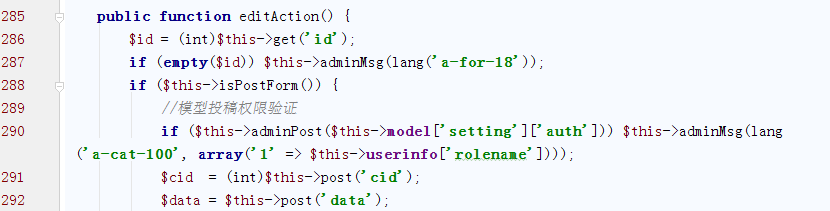
如果关联，获取关联表及记录（由此可见cid不是栏目）



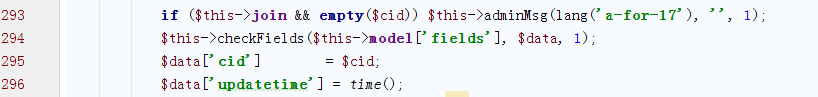
检查字段，设置属性，插入，生成静态文件，否则报错



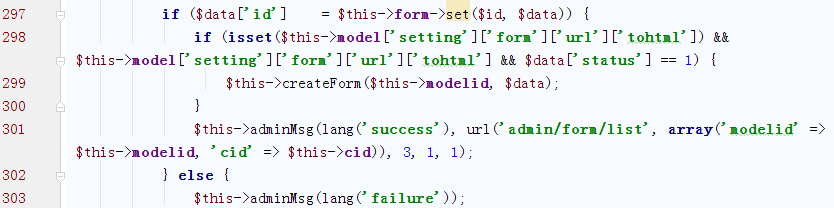
editAction



获取数据



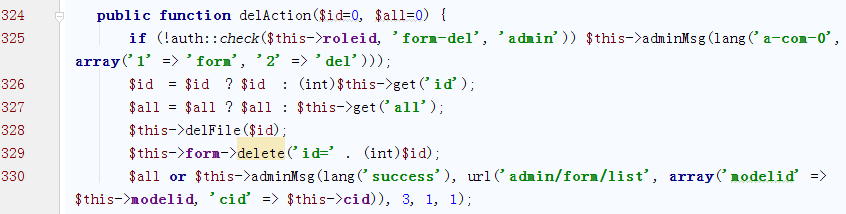
关联检查和检查字段



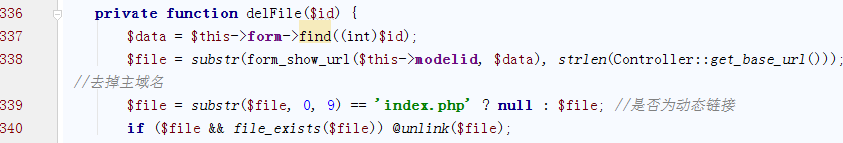
更新表，检查是否需要生成静态文件设置



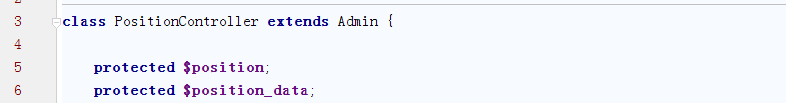
delAction



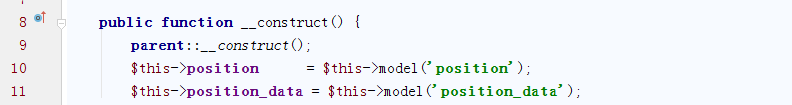
delFile



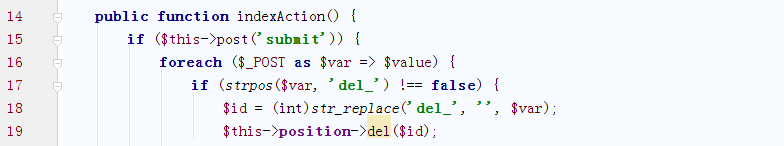
**PositionController**



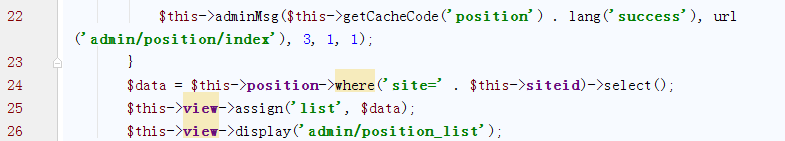
\_\_construct



indexAction



删除



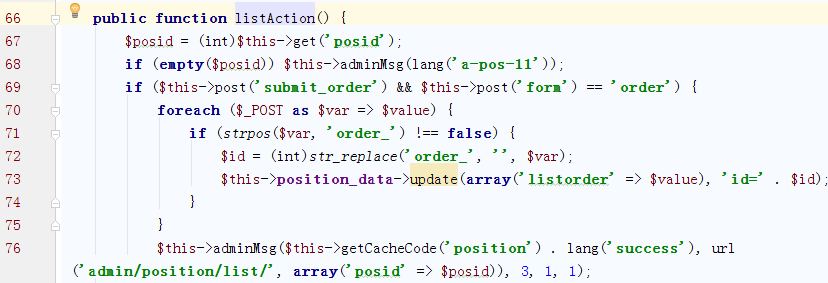
addAction

editAction

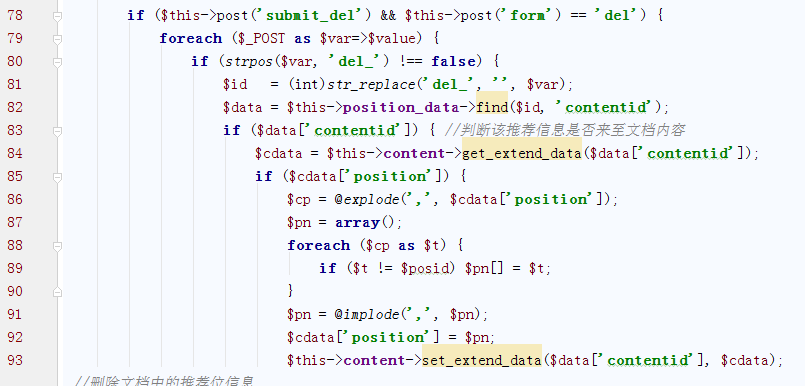
delAction

listAction

信息管理



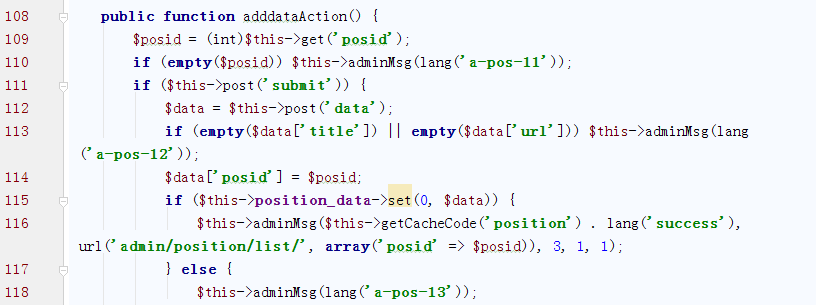
排序



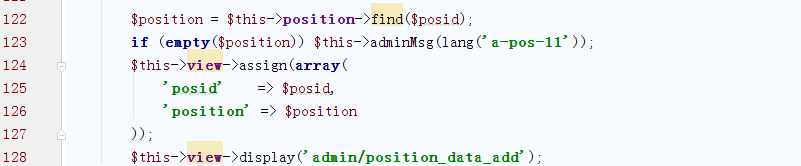
删除，判断是否来自文档模型，从content\_extend表中删除该位置



adddataAction

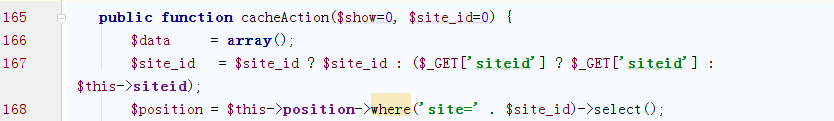


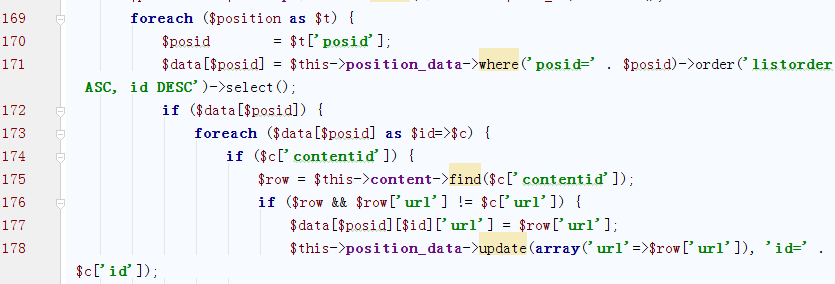
获取要添加数据的区域id（如在首页头条中添加），获取数据，添加到position\_data表中



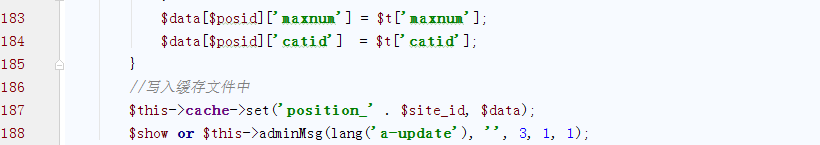
为啥添加完了后再检查区域块是否存在

cacheAction





获取区域块，遍历，从position\_data表中获取数据，如果为内容相关，获取内容的url等，并更新position\_data表



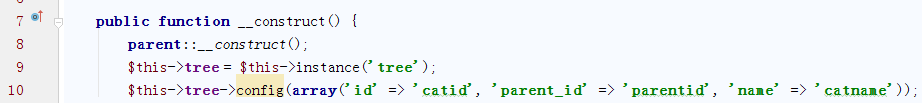
写入缓存

ajaxviewAction

ajaxloadinfoAction

CategoryController

\_\_construct



初始化tree

indexAction



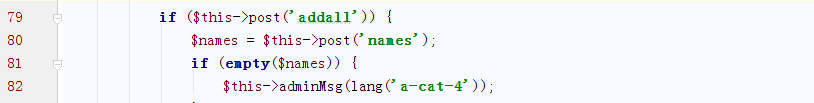
排序或删除

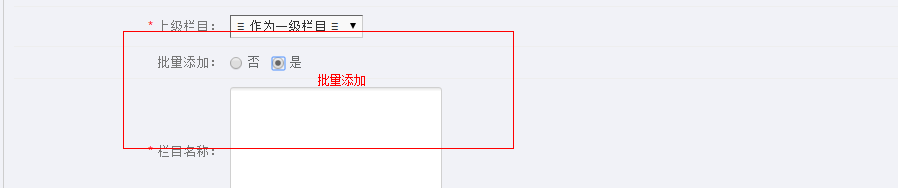


【为什么要清除表单模型所有的joinid】

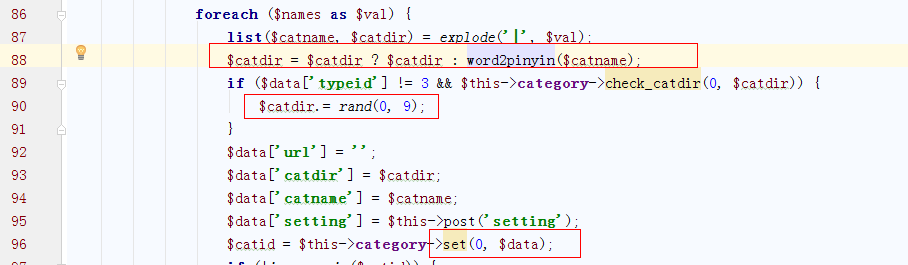
合并表单模型，将栏目整成树

addAction

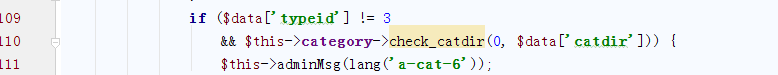




批量添加是按照【多个栏目直接使用回车符换行，格式：栏目名称|栏目目录[回车符]】形式添加



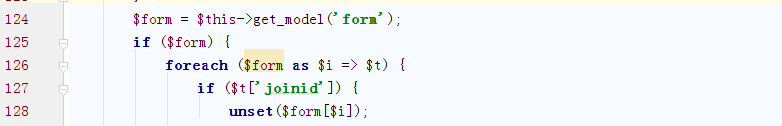
批量添加的目录（其实是所有目录）均为拼音，保存到category表中





栏目目录可能用于上传文件之类，check\_catdir检查是否存在，不存在返回false，因此就不会报错

另外，typeid为cat类别，3为单网页



此处的form好像最后模板中也没用到

editAction

delAction

urlAction

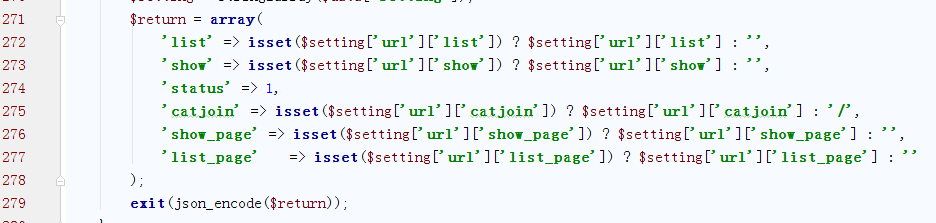




后台获取的就是所有设置



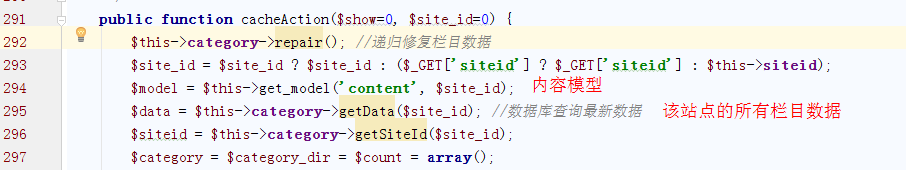
ajaximportAction



返回的就是配置中的url规则

cacheAction

【保存栏目信息，以栏目id为key，值有tablename，modelname，arrchilds，content，setting，pagesize，url，arrmodelid等，并更新原数据表的url等字段】



另外$siteid是该站点的继承站点（SITE\_EXTEND\_ID）

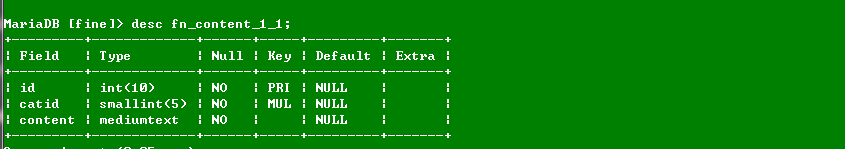


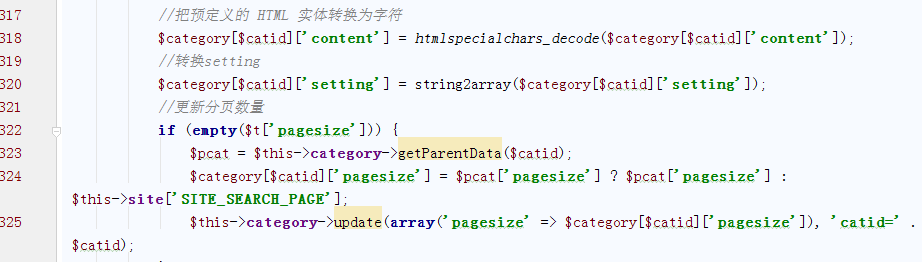


category表只保存栏目信息（单网页除外），因此，该栏目对应的数据得另外建1个表

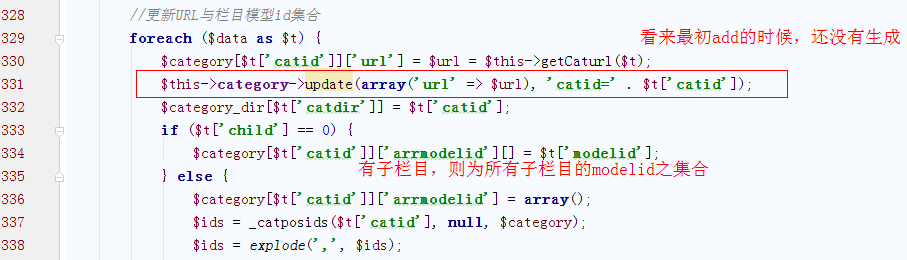


对应的model表里有该栏目对应的内容表，模板，配置等信息（貌似是一对一且不能修改）

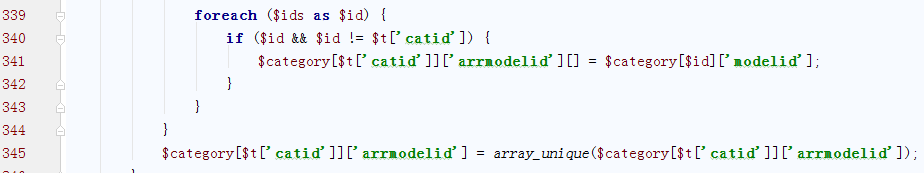




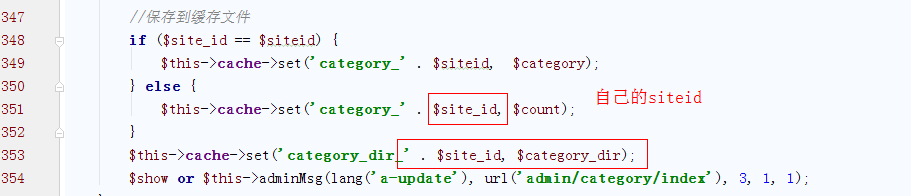
将内容转义，设置每页大小



获取所有子栏目的catid



设置modelid为所有子栏目的集合



如果没有继承siteid，则直接保存

如果有继承，只保存数量？？？

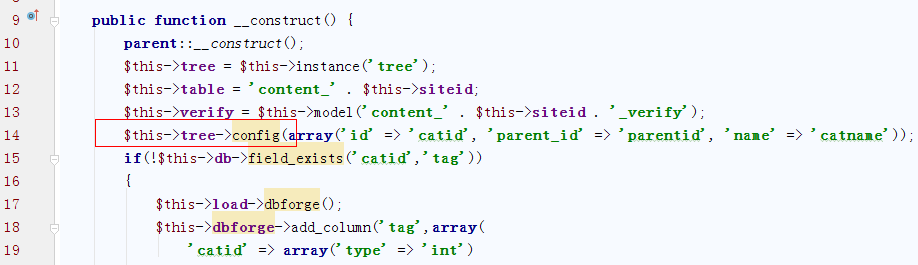
最后还保存栏目目录



就是目录名对应catid值

ContentController

\_\_construct



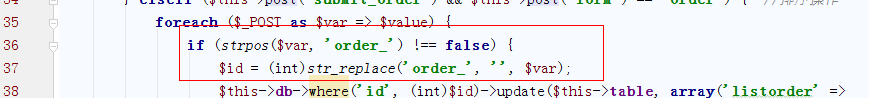
如果content表没有catid和tag字段，将其添加上

indexAction

（1）设置搜索参数

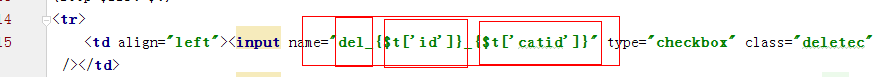
（2）排序



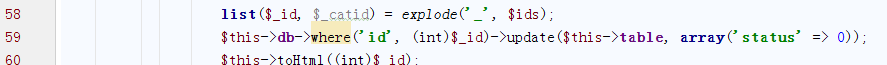


（3）标记通过





（4）标记回收站



（5）微信推送



（6）删除操作



（7）移动操作



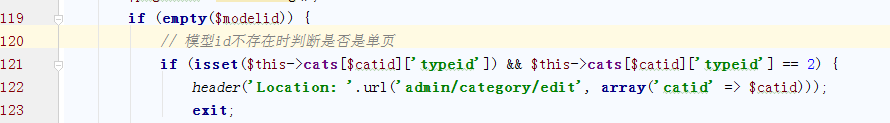
92行不是栏目模型，而是内容表的模型，101行更新的表的catid

100行更新的是content\_siteid表中的catid

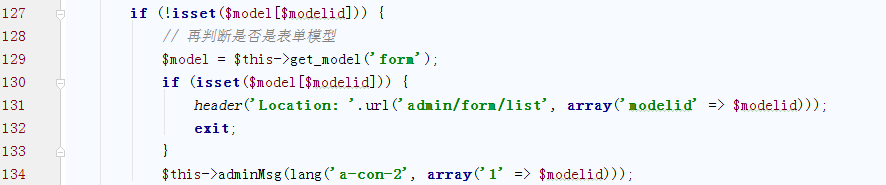
【更新cat对应的表模型中的catid可以理解，为什么还要更新content\_siteid表？？？】

（8）查询

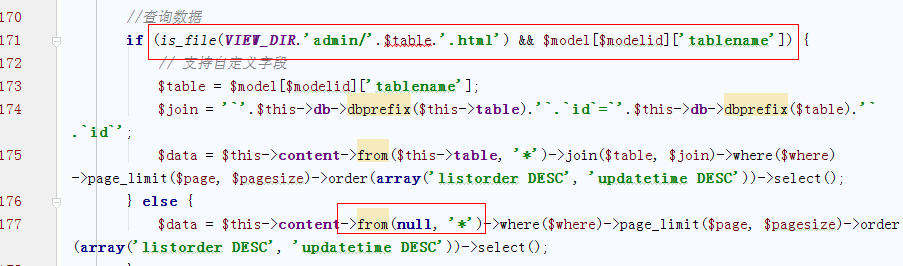




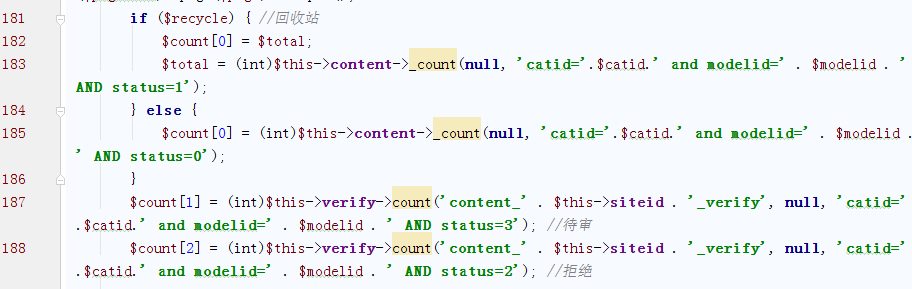
如果是单页模型，跳转到栏目编辑？？？？



表单模型（内容模型中没有），跳转到表单列表



要有表名对应的模板才能查询吗？





统计他们三的数量

verifyAction

editverifyAction



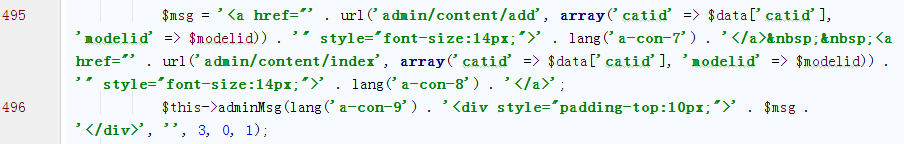
首先获取content\_verify表中的数据

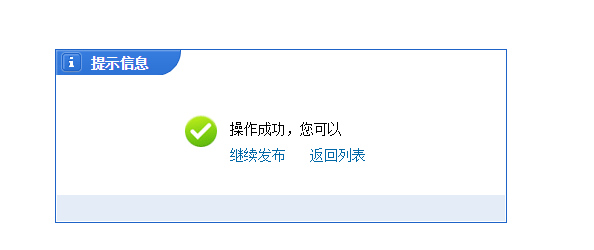


如果为修改

delverifyAction

addAction





getTag

按,分割标签

duplicateAction

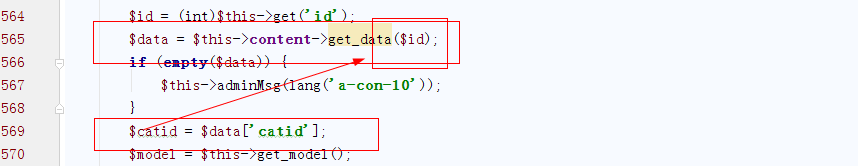


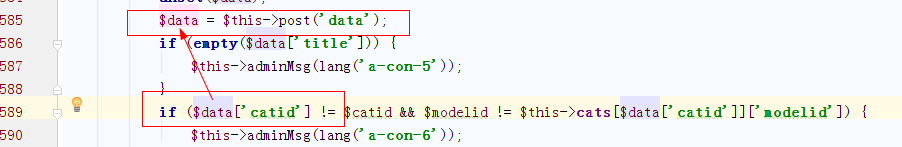
首先合并id对应的主附表数据





editAction





这里判断的是编辑的catid不能修改，必须是原文章的catid

delAction

删除

ajaxkwAction

获取关键字

ajaxtitleAction

标题是否重复检查

ajaxloadinfoAction

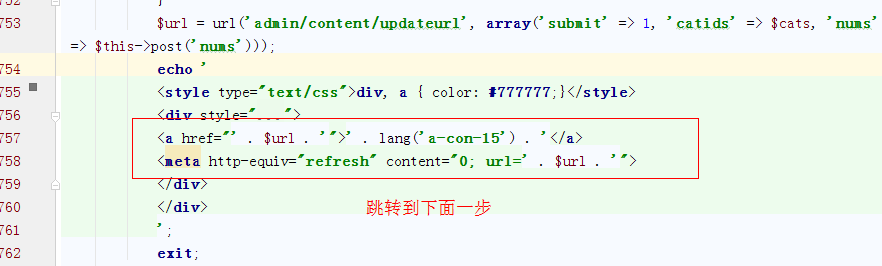
当进行关联文章、推荐位操作时，加载content表文章列表

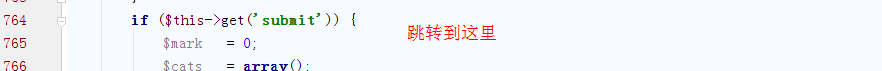
updateurlAction

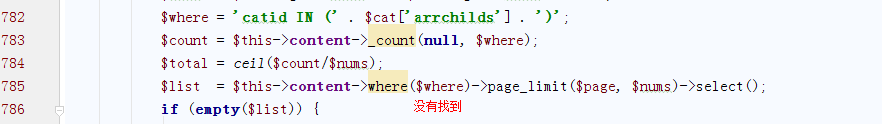
更新内容url









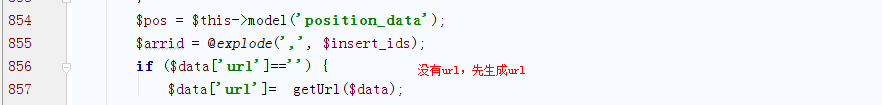




toHtml

调用createShow生成/删除内容页HTML文件

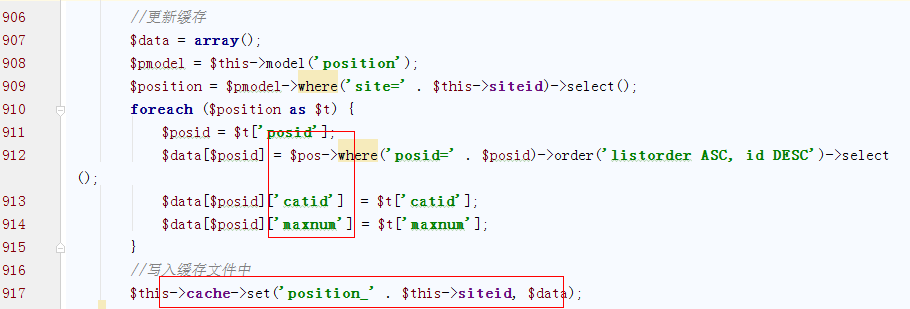
setPosition





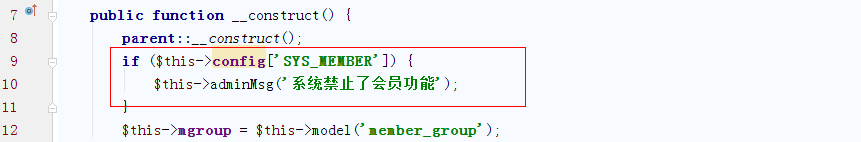




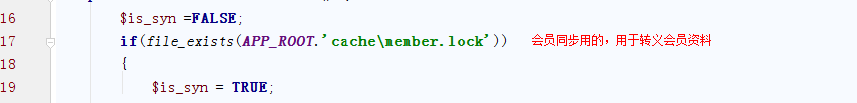


MemberController

\_\_construct



indexAction



（1）获取搜索关键字

（2）标记审核

（3）标记未审

（4）搜索

groupAction

（1）添加会员

（2）编辑

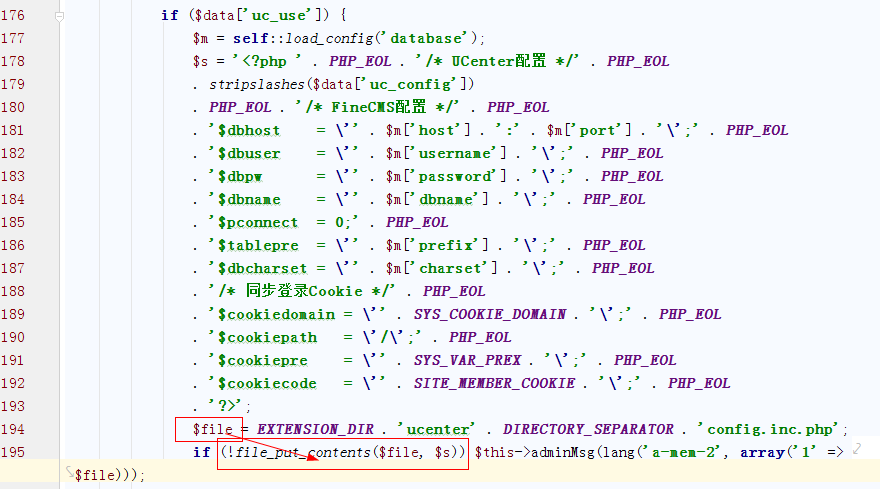
（3）生成会员组缓存

（4）删除

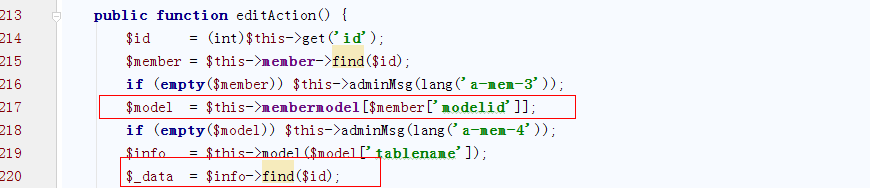
（5）排序

（6）批量删除

configAction



editAction



会员也有模型，member表有modelid字段，对应model表里的模型







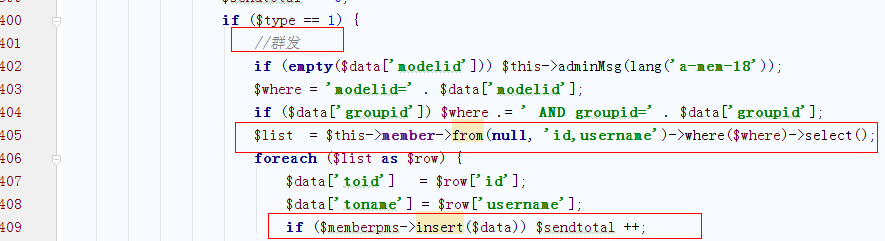
member表和模型表都更新

regAction

pmsAction

（1）显示某条短消息内容，包括删除

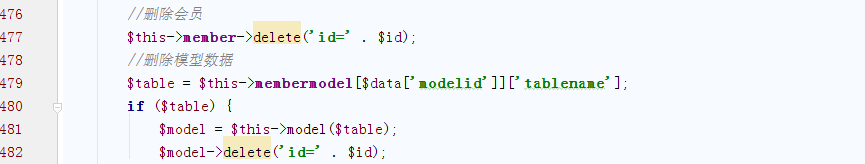
（2）发送系统消息



modelid应该是默认，从member表中获取id和name，然后添加到memberpms表中

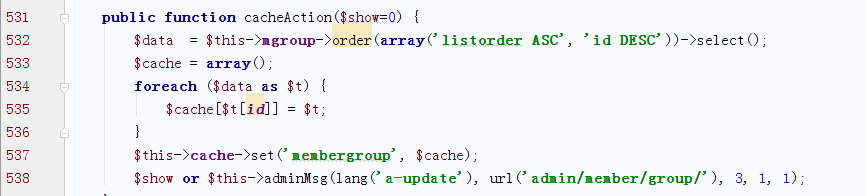
（3）显示短消息列表

delAction





cacheAction



将会员组数据，以组id为key存为缓存

extendAction



貌似是这个样子，不知道哪里调用

testAction

测试链接数据库

ajaxemailAction

ajaxusernameAction

setspace

UserController

synAction

貌似同步用户数据

indexAction

addAction

增加一个用户（检查用户名是否重复）

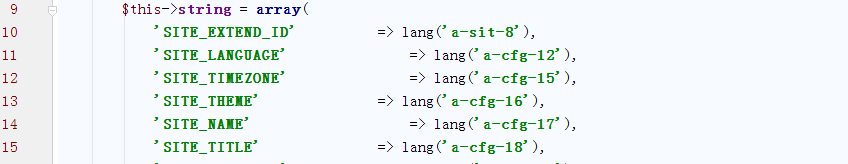
editAction

ajaxeditAction

delAction

SiteController

\_\_construct



定义网站设置的数组（别看是大写，其实是字符串）

indexAction

站点列表（不用）

addAction

添加站点（不用）

editAction

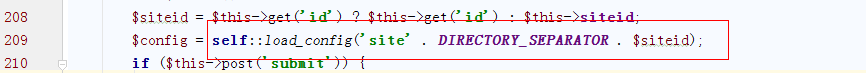
修改站点（不用）

delAction

删除站点（不用）

configAction

站点配置



获取已有配置

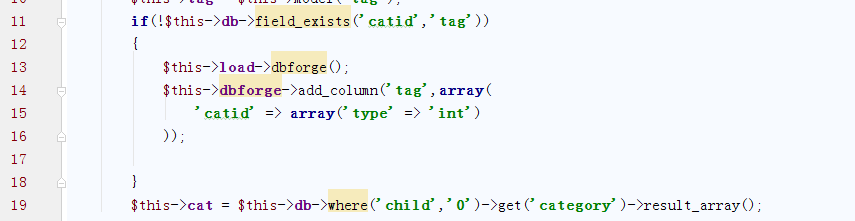


string就是构造函数里的那个数组

TagController

【tag是关联到cat的？？？？？】

\_\_construct



tag表如果没有tag、catid字段，添加

获取child=0的所有栏目

indexAction

删除、更新（查重）、查询

addAction

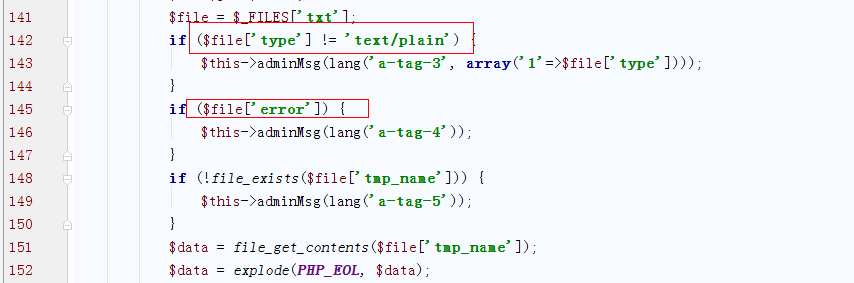
editAction

delAction

importAction

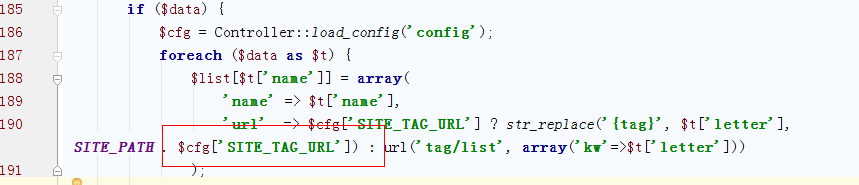
导入





之前没见过

cacheAction



？？？？？？？？？？？？？tagurl规则，貌似点击tag，会跳转到涉及该tag的所有文章

HtmlController

生成静态页面，保存到cache/views文件夹下（用不着）