**App**

**1.整体功能**

均为静态方法，类似核心文件，提供自动加载、实例化核心model、加载配置等功能

**2.属性**

$namespace目录

$controller

$action

$plugin

$siteid网站id

$config配置项

$language语言项（已包含语言文件中的内容）

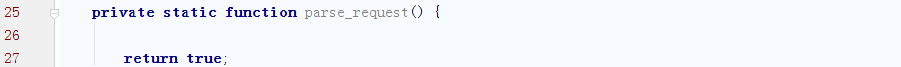
$\_objects

$site\_info网站配置

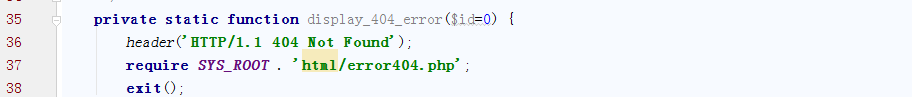
$\_inc\_files

**3.方法**

parse\_request

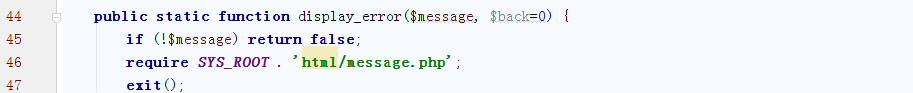


display\_404\_error



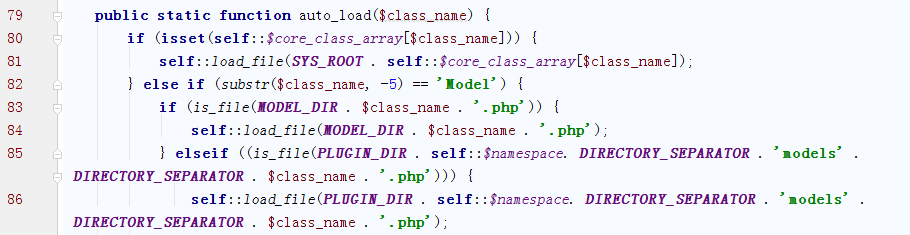
引入error404模板

display\_error



如果有消息，引入message模板

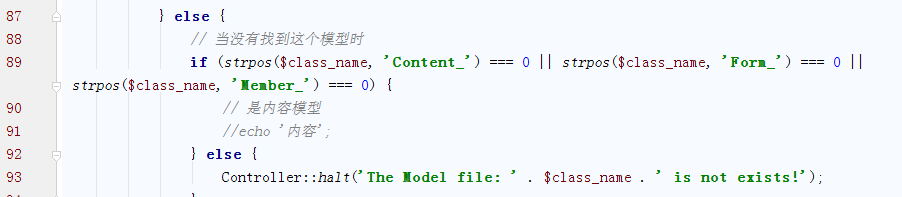
auto\_load



如果在$core\_class\_array核心类引导数组中，直接安装对应的路径载入

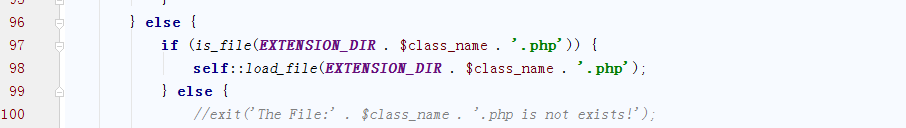
|  |
| --- |
| 都是libraries中的一些lib文件 |

如果类名为model，按model文件夹、插件文件夹顺序载入



如果没找到，如果为content、form、member类，不做处理

如果不是，报错



如果extension文件夹中存在，载入

（看来顺序是：引导数组、模型、模型插件、extension）

get\_namespace\_id等







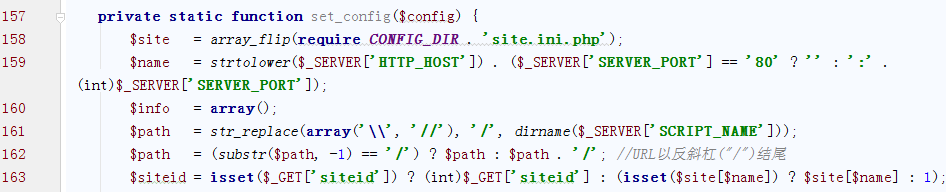




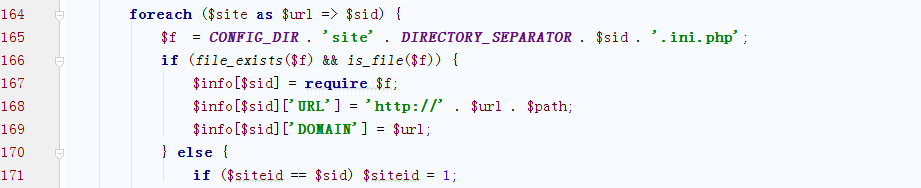




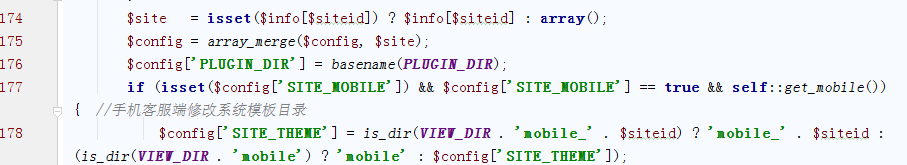
set\_config



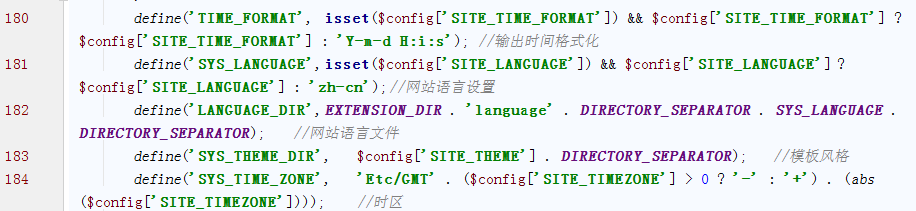
交换site.ini.php中配置的key-value值，获取访问的域名，获取url路径，获取siteid（根据get和访问域名判断）



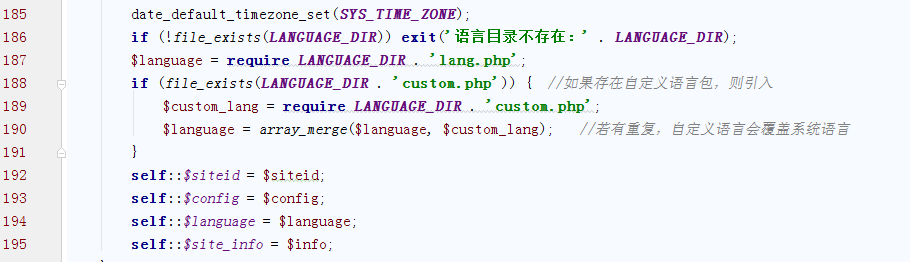
获取site对应的配置，组装url、domain，或设为默认值1



合并配置，设置插件目录，如果开启移动访问，且当前为移动访问，设置网站模板为对应的移动文件夹

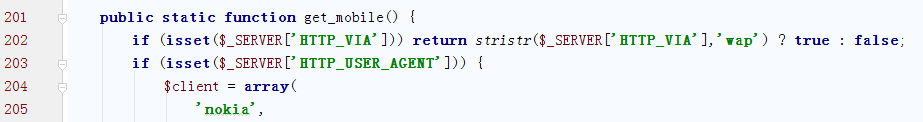


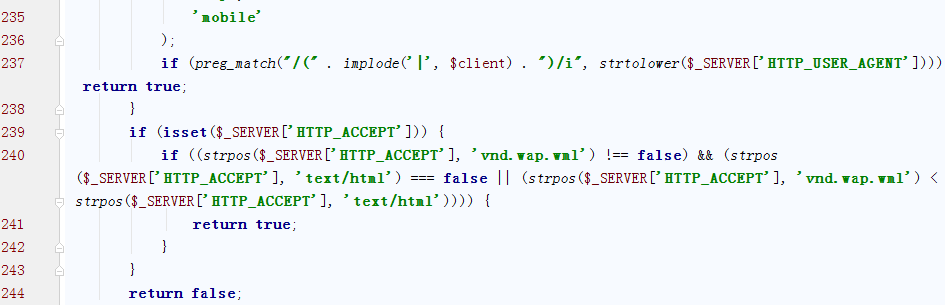
设置时间格式、系统语言路径、模板风格、时区



设置时区，检查语言目录是否存在，将以上值保存包属性中

get\_mobile



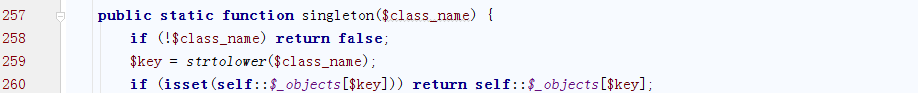


检查HTTP\_VIA、HTTP\_USER\_AGENT，HTTP\_ACCEPT等，判断是否为手机访问

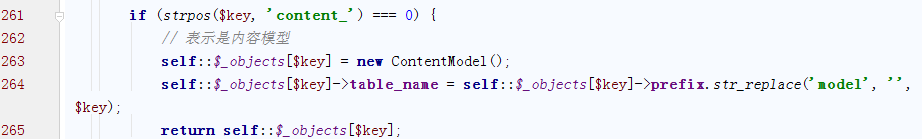
get\_plugin\_id



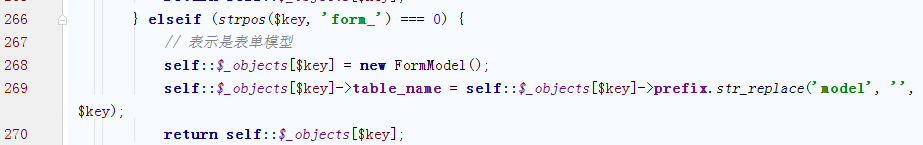
singleton



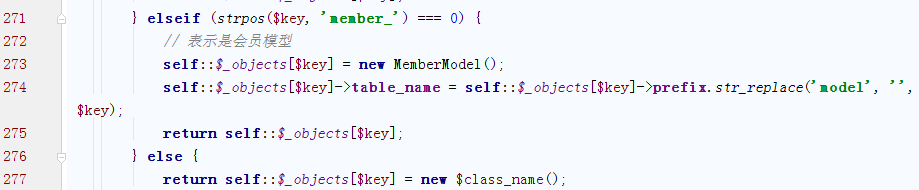
判断类是否已实例化



如果是内容，实例化ContentModel

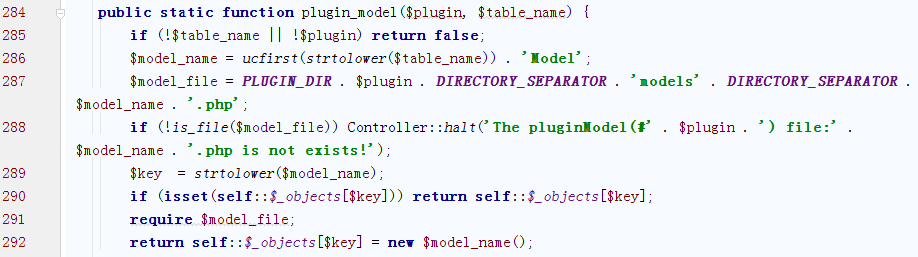


如果是表单，实例化FormModel



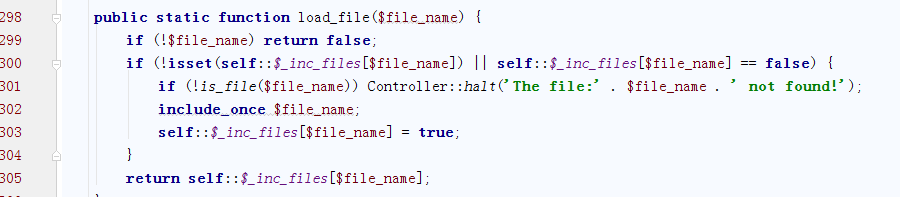
如果是会员，实例化MemberModel，否则直接实例化

plugin\_model



获取对应的模型名称，模型路径，检查是否已实例化，引入，实例化

load\_file

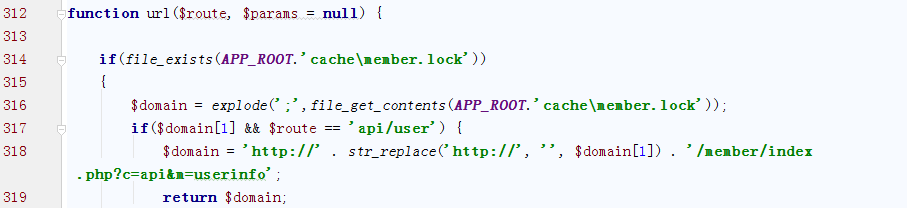


检查是否已引入，引入

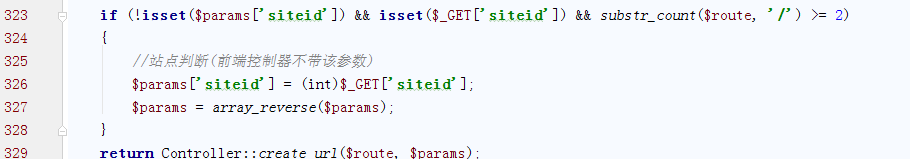
**以下不是APP类的成员方法，而是全局函数**

url

目的是将/形式的url转为query形式的url



如果存在会员锁文件(可能和用户自定义域名有关)，获取，检查domain和route是否为api，是则返回该api



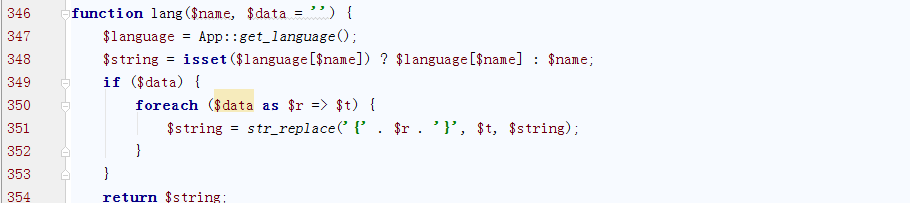
如果入参没有siteid但get中有，且存在访问路径/，将get中的siteid赋给params，反置，调用Controler的create\_url创建url





在route前加入目录id

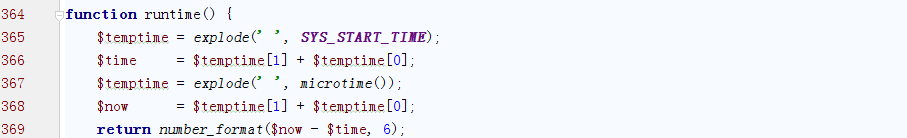
lang



获取当前语言文件，检查对应的语言项是否存在，并data替换其中的占位符



runtime



计算运行时间



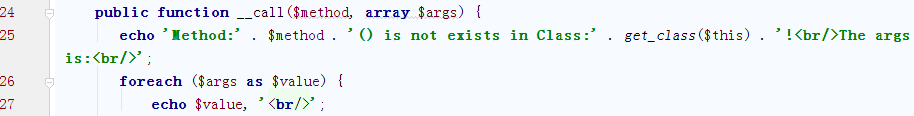
注册自动加载函数

**Fn\_base**

全部为魔术方法，为根基类

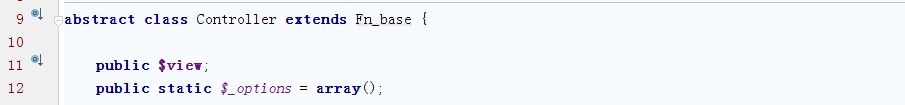








**Controller**

****

1.整体功能

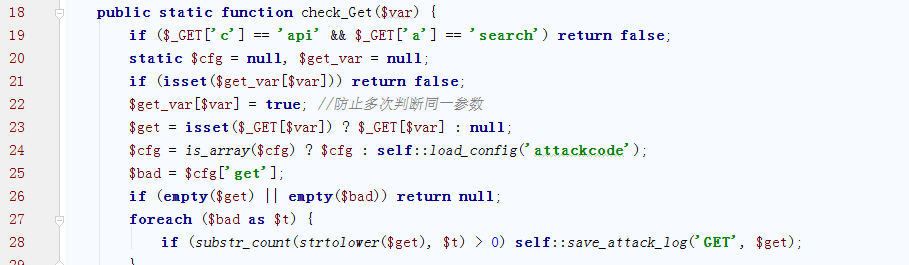
主要有检查攻击、创建url等功能，全是静态方法，类似于工具类

2.属性

基类中没有用到属性

3.方法

check\_Get



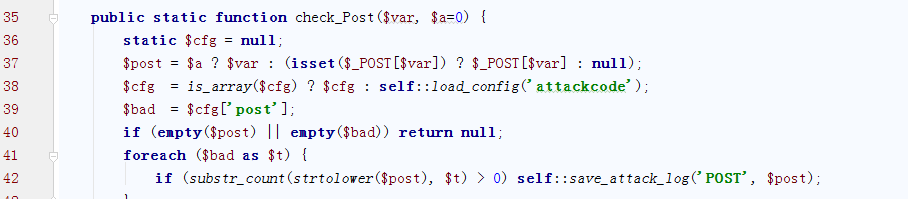
如果访问api或为搜索，不检查

如果静态属性$get\_var中已检查过（本次访问已检查过就不检查了，也不计入到日志中），不检查

载入attackcode配置文件到静态属性$cfg中，遍历，如果存在，记录到攻击日志中

（貌似没有返回？）

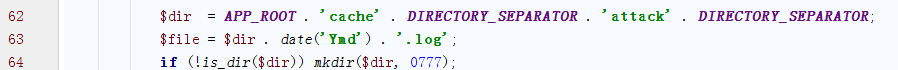
check\_Post



save\_attack\_log



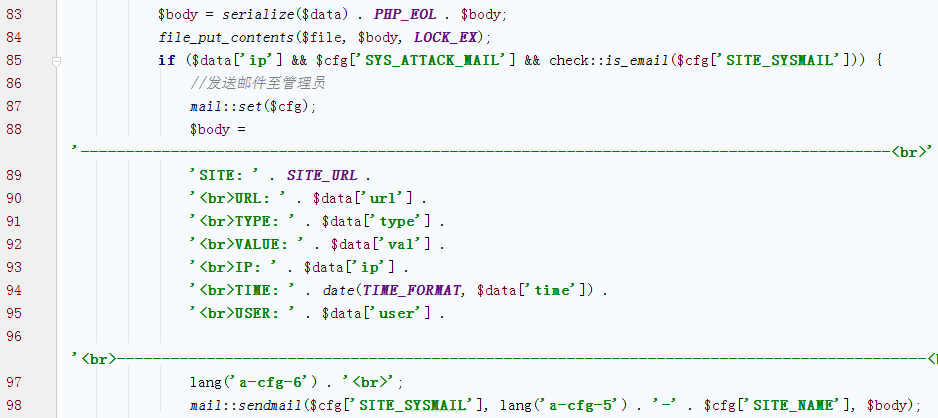
如果开启攻击日志记录，去掉uri中的域名部分，设置访问信息



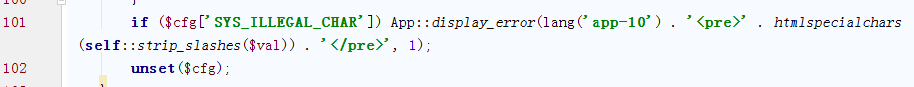
设置攻击日志目录，文件名，如果没有则创建



如果文件存在，获取文件内容，遍历，根据ip检查攻击次数，如果大于，报错

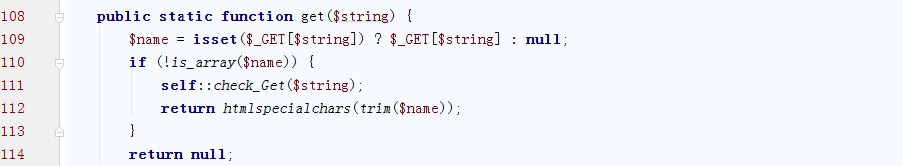


将本次攻击组合包之前文件内容前面，写入，如果开启攻击邮件提示，通知管理员



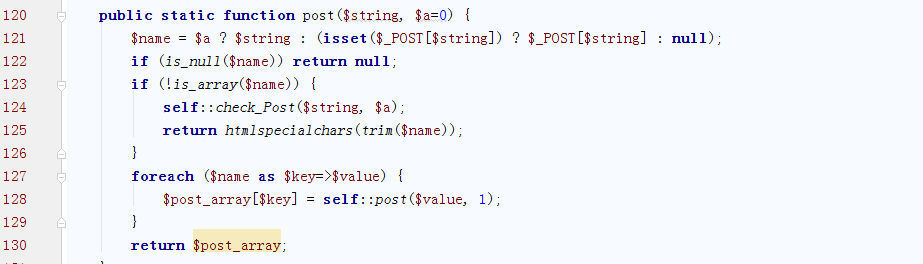
如果开启禁止非法字符提交，报错

get



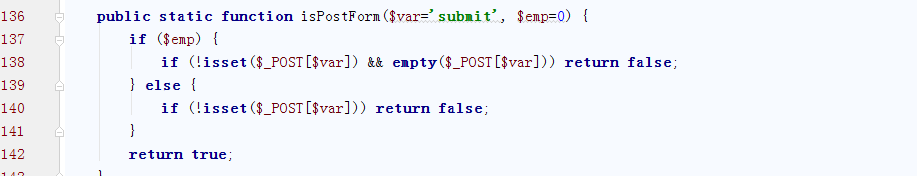
获取并分析$\_GET数组某参数值，不能是数组

post



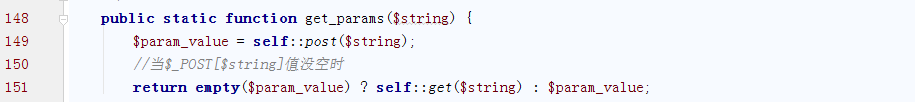
可以是数组

isPostForm



验证表单是否POST提交

get\_params



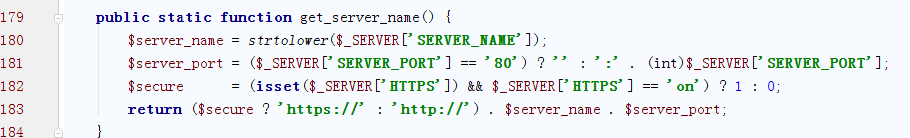
按$\_POST和$\_GET顺序获取某参数的值

halt



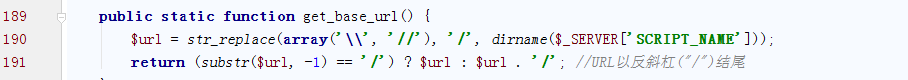
获取报错时的文件，行数，参数等信息，引入exception.php文件，写入日志

get\_server\_name



获取主机名，组装完整的主机域名

get\_base\_url



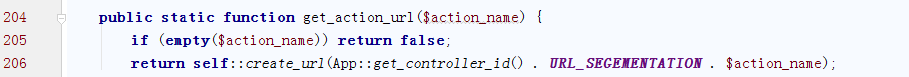
获取跟域名

get\_self\_url

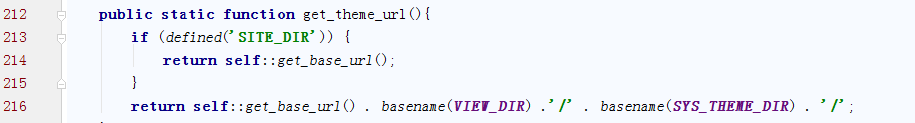


创建当前运行的Action的url

get\_action\_url



get\_theme\_url



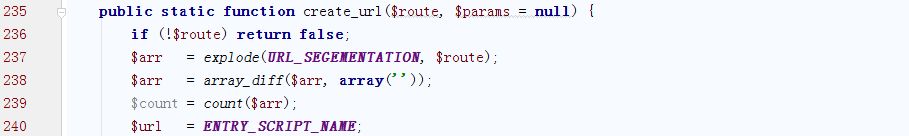
获取当前项目themes目录的URL

redirect

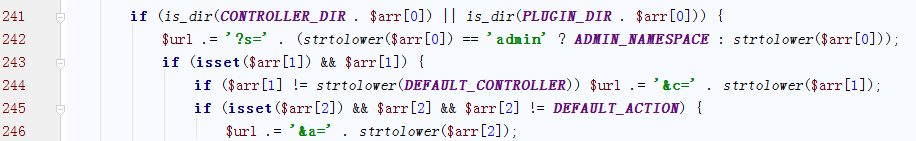


网址(URL)跳转操作，先判断是否已发送header，有则用js跳转

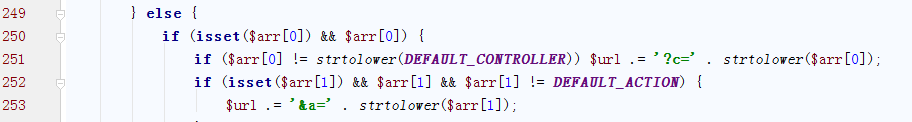
create\_url



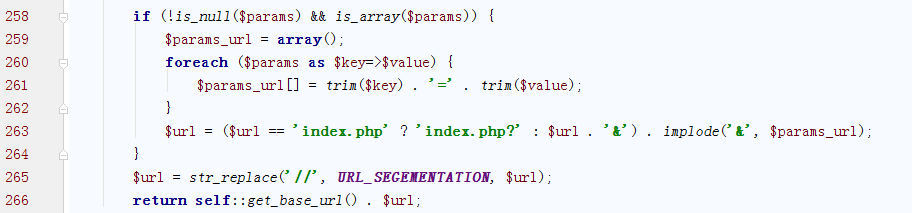
先分割route，获取入口文件



如果访问路径在普通路径或者插件路径为目录（只能是admin或者插件才是目录），如果为admin（？插件呢？），直接组装（判断c、a是否为默认值，默认值不用组装）

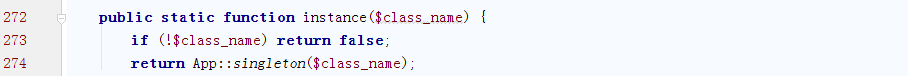


如果route有值，检查是否不为/，不为则组装为c，a等

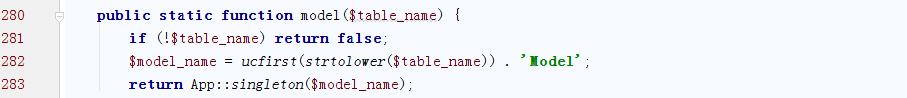


拼接后面的query

instance



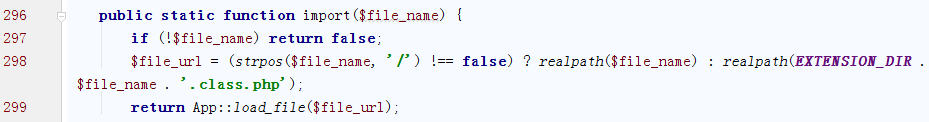
model



plugin\_model



import



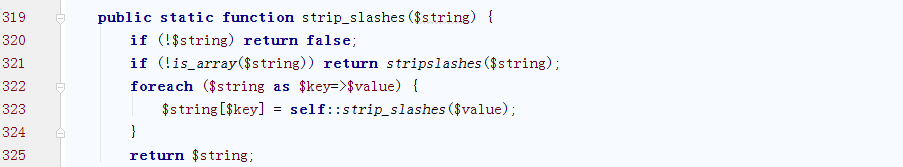
先获取完整路径再引入？

load\_config

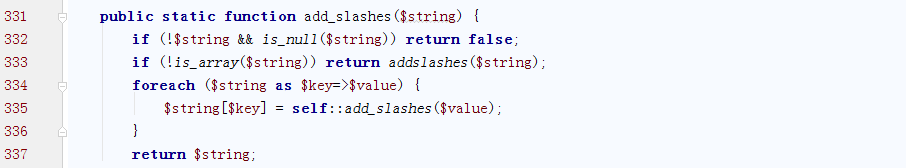


如果未引入，则引入

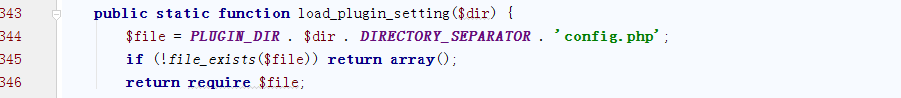
strip\_slashes



add\_slashes

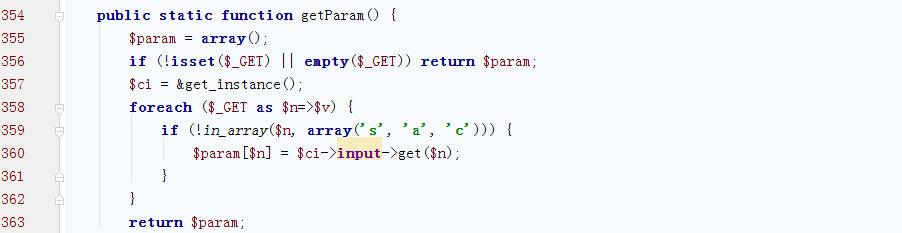


load\_plugin\_setting



获取插件中的配置

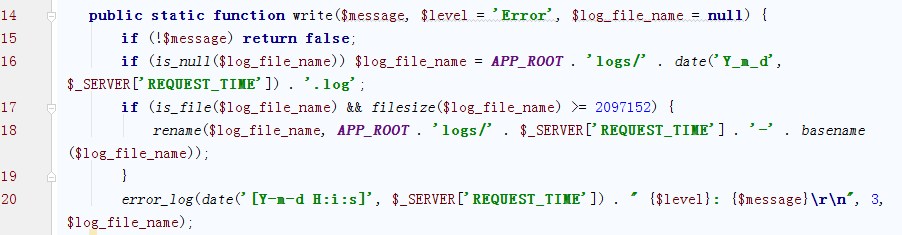
getParam



获取不为s、a、c的参数

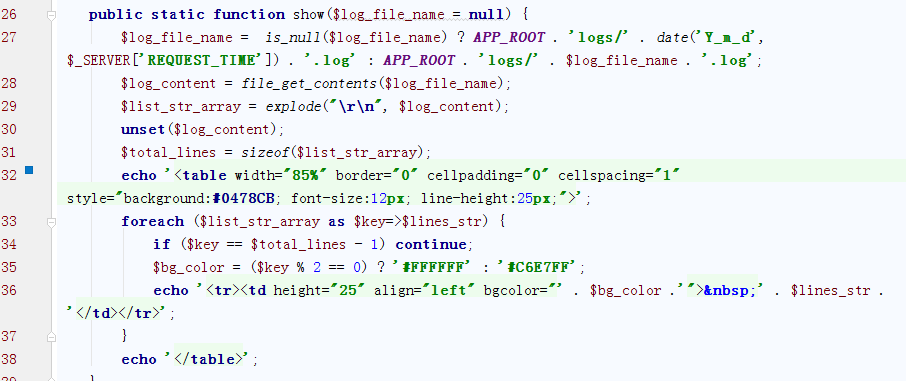
**Log**

write



如果没有指定日志文件，则为默认的日志文件，调用error\_log写日志（文件重命名？）

show



获取日志文件内容，输出为表格形式