## 从开始到结束所被定义过变量;

CHARSET : utf-8

DOC\_ROOT : str\_replace("\\", '/', dirname(\_\_FILE\_\_) ).'/'

APP\_PATH: DOC\_ROOT.'system/'

APP\_DEBUG : false

IN\_APP : true

APP\_ROOT : str\_replace("\\","/",substr(dirname(\_\_FILE\_\_), 0, -6))

LIB\_PATH : APP\_PATH.'library/'

CORE\_PATH : APP\_PATH.'core/'

CONF\_PATH : DOC\_ROOT.'config/'

CACHE\_PATH : DOC\_ROOT.'caches/'

TPL\_PATH : DOC\_ROOT.'template/'

LANG\_PATH : APP\_PATH.'language/'

\_\_APP\_\_ : $\_SERVER['SCRIPT\_NAME']

\_\_ROOT\_\_ : str\_replace(basename(\_\_APP\_\_), "", \_\_APP\_\_)

PLUGIN\_PATH : APP\_PATH.'plugin/'

IS\_CGI : (0 === strpos(PHP\_SAPI,'cgi') || false !== strpos(PHP\_SAPI,'fcgi')) ? 1 : 0

IS\_WIN : strstr(PHP\_OS, 'WIN') ? 1 : 0

IS\_CLI : PHP\_SAPI=='cli'? 1 : 0

EXT : '.class.php'

MAGIC\_QUOTES\_GPC : function\_exists('get\_magic\_quotes\_gpc') && get\_magic\_quotes\_gpc()

REQUEST\_METHOD : $\_SERVER['REQUEST\_METHOD']

IS\_GET : REQUEST\_METHOD =='GET' ? TRUE : FALSE

IS\_POST : REQUEST\_METHOD =='POST' ? TRUE : FALSE

IS\_PUT : REQUEST\_METHOD =='PUT' ? TRUE : FALSE

IS\_DELETE : REQUEST\_METHOD =='DELETE' ? TRUE : FALSE

TIMESTAMP : time()

IS\_AJAX : // 太长了, 具体描述的是当前是否是ajax 请求

MODULE\_NAME : //你可以认为是URL里面的m参数的值,如果m没值默认是goods

CONTROL\_NAME : //你可以认为是URL里面的c参数的值,如果c没值默认是index

METHOD\_NAME : // 你可以认为是URL里面的a参数的值,如果a没值默认是index

MODULE\_PATH :

str\_replace(DOC\_ROOT, '', APP\_PATH.config('DEFAULT\_H\_LAYER').'/'.$module\_name.'/')

以上变量都是通过 define 定义的.也就是说在当前全局都可以方法得到.

注意,以下所获得的值都是通过配置得到的,具体请看以下方法:

MODULE\_NAME: 在system/core/hd\_application.class.php --> \_getModule()

CONTROL\_NAME: 在system/core/hd\_application.class.php --> \_getControl()

METHOD\_NAME 在system/core/hd\_application.class.php --> \_getMethod()

## 当连接进来后服务器所经过的流程:

### 1.介绍入口:

整个网站的唯一入口都是index.php文件.

它并不是通过文件名来分辩你访问的是那个页面 , 是通过URL参数来判断你当前访问的是哪个页面.请看下图



是通过 m,c,a 这三个参数(参数名可配置,不一定是 m,c,a名称,而且m,c,a可以省略).

看这链接你可以这样认为到install(m)目录找index(c)文件,执行里面的index(a)方法.

### index.php所做的那些事

Index.php在执行的时候引用了base.php.

Base.php做了以下比较重要的事:

1).引入了 hd\_core.class.php和function.php 文件.

--hd\_core.class.php是负责处理全局事物的.

--function.php 是封装了全局公用的方法.

2).把 hd\_core.class.php 里面的 handleException 方法注册为全局异常捕抓

3).把 hd\_core.class.php 里面的 handleError方法注册为全局错误捕抓

4).把 hd\_core.class.php 里面的 handleShutdown方法注册为流程结束触发

5).把 hd\_core.class.php 里面的 autoload方法注册到\_\_autoload.

注意,autoload在引用类文件的不相同处:

如果所引用的文件是以 hd\_ 开头的就到 system/core 目录下面去找

如果不是hd\_ 开头并且也没包含control 字符就到 system/library 路径下去找

如果所引用的文件名存在 control 字符就执行以下:

假设 $class 变量现在值为 init\_control

(以下所有的动作都在hd\_core.class.php文件中完成)

1. .
2. .load\_class 里面所执行最重要的方法



然后就是 拼出文件的全路径



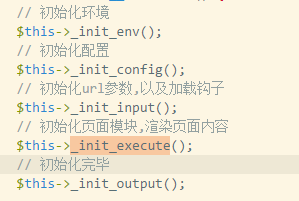
最后就剩引用了

路径例子: E:\Apache24\htdocs\system\module\goods\control\init\_control.class.php

6).执行 run() 方法,创建出 hd\_application.class.php 对象, 该文件是真正的程序处理,是m,c,a 值得处理地方.

### hd\_application.class.php 所做的那些事

在创建 hd\_application.class.php 对象的时候会执行 以下5个步骤



下面一一对这5步进行细讲:

**1).\_init\_env() 这个内容比较简单,就不细讲:**

**2).\_init\_config() 这个内容比较简单,就不细讲:**

**3).\_init\_input()**

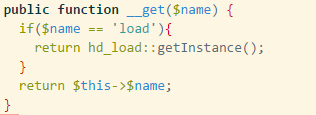
主要方法如下:



这里主要讲 \_hook\_register()



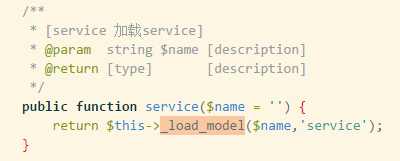
关于 $this->load load这个变量在类中没定义,却可以使用? 关键在于它所继承的hd\_base.class.php 在该父类中有这么一段代码:



想必知道怎么回事了

下面进入一个比较关键的一点  这里面都做了什么事情.我们先进入 hd\_load.class.php 这个文件

看以下代码段:



这里它调用了自身的\_load\_model方法. 而这个方法主要是加载文件.他有两种加载方式.第一种: 加载非插件.第二种加载插件.下面来分析下加载过程.

**非插件形式:** **假如 $name =** **'****admin/****app' $layer =** **'****service'**

首先按照name的值分别解析成以下变量

list($module, $name) = explode("/", $name);

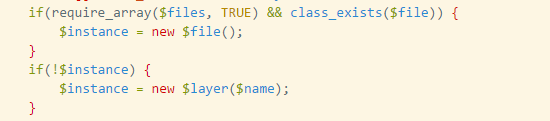
$file = 'app\_service'

$class = 'app\service'

$files[] = '/system/module/'.$module.'/model/'.$layer.'/'.$file.EXT;

// system/module/admin/model/service/app\_service.class.php

然后加载这个文件,new出对象



将new的对象缓存起来,然后把对象返回出去



**插件形式:假如 $name = '#****hello#admin/app' $layer = 'service'**

首先按照name的值分别解析成以下变量

list(, $pluginid, $name) = explode("#", $name);

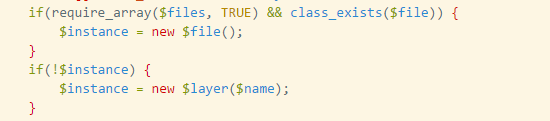
$file = 'app\_service'

$class = 'app\service'

$files[] = '/system/plugin/'.$pluginid.'/model/'.$layer.'/'.$file.EXT;

// '/system/plugin/hello/model/service/app\_service.class.php'

然后加载这个文件,new出对象



将new的对象缓存起来,然后把对象返回出去



关于 app\_service.class.php 这个文件做了什么,还是需要自己去看看



这段调用了 app\_service.class.php 里面的 get\_apps() 方法获取所有钩子,然后把这些钩子注册到全局钩子中, 什么时候触发钩子下面的下面会讲到.

**3).\_init\_execute()**

主要:



根据 url里面的 m 以及 c 两个值进行加载类文件以及创建出对象.

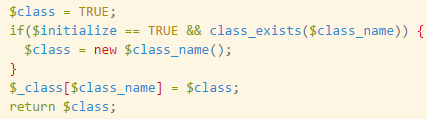
现在进入 hd\_core.class.php 下面 load\_class 方法里面去看看

假设: $module = install $class = index

$class\_name = index\_control

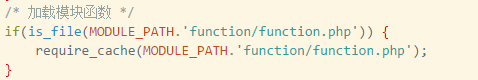
$class\_dir = '/system/module/install/control/'

$class\_file = $class\_dir.$class\_name.EXT



然后返回创建对象.

默认加载:

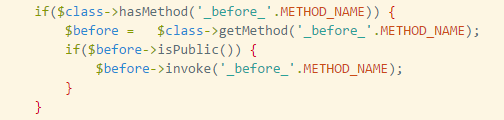


执行方法:

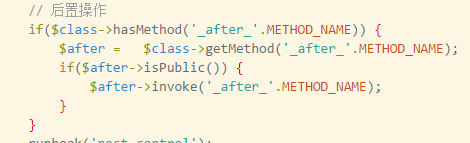


在做这个执行这个方法之前会有两个步骤可做

在执行方法之前:



以及在执行方法之后



好的,基本流程就这样,结束!

## 关于钩子:

### 钩子数据结构:

所有注册的钩子都会被放到 hd\_hook.class.php 文件里面的 静态变量 $hooks 中

数据结构是,例子:

[

'pre\_system': ['E:\Apache24\htdocs\system\library\hook.class.php'],

'post\_control':[

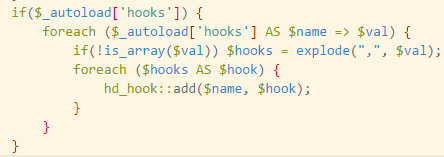
'E:\Apache24\htdocs\system\library\hook.class.php', 'E:\Apache24\htdocs\system\library\post.class.php'

]

]

### 在这几个步骤里面钩子被注册到系统里面:

1). hd\_application.class.php --> \_init\_config() 方法里面的



2). hd\_application.class.php --> \_hook\_register() 方法里面的



### 钩子的触发阶段:

在 hd\_application.class.php 的 \_init\_execute() 方法里面执行了三次 runhook()方法.

Runhook 这个方法是触发钩子的关键.

关于钩子触发阶段有以下.

hd\_application.class.php -> \_init\_execute()

runhook('pre\_system');

runhook('pre\_control');

runhook('post\_control');

hd\_view.class.php -> get()

runhook('pre\_response');

-> display()

runhook('pre\_output',$this->data);

System/function/function.php -> pages()

runhook('before\_pages\_compile',$string);

Form.class.php -> editor()

runhook('editor',$data = array('value'=>$value,'string'=>&$string,'upload\_init'=>$upload\_init));

Input.class.php -> data()

runhook('pre\_input');

Template.class.php -> compile()

runhook('tmpl\_compile',$template);

runhook('after\_tmpl\_compile',$template);

-> hooktags()

Upload.class.php -> upload()

runhook('before\_upload\_img',$file['url']);

runhook('before\_upload\_img',$file['url']);

-> remote()

runhook('before\_upload\_img',$file['url']);

site\_control.class.php -> reg()

runhook('before\_set\_reg');

admin\_group\_service.class.php -> save();

runhook('admin\_group\_save',$data);

admin\_service.class.php -> init()

runhook('admin\_init',$\_admin);

-> login()

runhook('admin\_login',$admin\_user);

cloud\_service.class.php -> bind\_site()

runhook('update\_cache');

太多了,写不下去了,自行全局查找吧!