|  |
| --- |
| 广州市艮车贸易有限公司 |
| CXSV框架使用说明文档 |
|  |
|  |
| **sang** |
| **2015/4/22** |

|  |
| --- |
|  |

Csv框架使用说明v1.0

**\*版本信息&保密等级**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **更改日期** | **更改要点说明** | **编制** | **审核** | **批准** |
| V1.0 | 2015/04/22 | 新建 | sang |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 文档保密等级： | | □机密 ■内部 □公开 | | | |

目录

[Csv框架使用说明v1.0 1](#_Toc417482088)

[1 引言 4](#_Toc417482089)

[2 框架开发规范 4](#_Toc417482090)

[2.1 控制器Action编写规范 4](#_Toc417482091)

[2.2 控制器与访问路径的关系 5](#_Toc417482092)

[2.3 模型Model编写规范 5](#_Toc417482093)

[3 目录结构 6](#_Toc417482094)

[3.1 应用根目录 6](#_Toc417482095)

[3.2 应用目录结构 6](#_Toc417482096)

[3.2.1 不分前后台的情况下的目录结构 6](#_Toc417482097)

[3.2.2 分前后台的情况下应用的目录结构 7](#_Toc417482098)

[3.3 命名空间对应目录 8](#_Toc417482099)

[3.3.1 Lib 8](#_Toc417482100)

[3.3.2 Mod 8](#_Toc417482101)

[3.3.3 App 8](#_Toc417482102)

[3.3.4 Act 8](#_Toc417482103)

[3.3.5 Widget 8](#_Toc417482104)

[3.3.6 Task 8](#_Toc417482105)

[4 注意事项 9](#_Toc417482106)

[4.1 关于exit和die退出语句的使用 9](#_Toc417482107)

[4.2 关于header函数 9](#_Toc417482108)

[4.3 关于static关键字 9](#_Toc417482109)

[4.4 关于cookie 9](#_Toc417482110)

[4.4.1 cookie函数的使用 9](#_Toc417482111)

[4.5 关于session 10](#_Toc417482112)

[4.5.1 session函数的使用 10](#_Toc417482113)

[4.6 关于储存storage 10](#_Toc417482114)

[4.6.1 storage函数的使用 10](#_Toc417482115)

[4.7 关于配置项config 11](#_Toc417482116)

[4.7.1 config函数的使用 11](#_Toc417482117)

[4.8 如何获取一个模型 model 12](#_Toc417482118)

[4.8.1 model函数的使用 12](#_Toc417482119)

[4.9 关于缓存cache 12](#_Toc417482120)

[4.9.1 cache函数的使用 12](#_Toc417482121)

[5 开发示例 13](#_Toc417482122)

[5.1 数据的增删改查 13](#_Toc417482123)

[6 附录 13](#_Toc417482124)

[6.1 常用的一些公共函数 13](#_Toc417482125)

[6.1.1 get 13](#_Toc417482126)

[6.1.2 post 14](#_Toc417482127)

[6.1.3 request 14](#_Toc417482128)

[6.1.4 add\_action 14](#_Toc417482129)

[6.1.5 apply\_action 14](#_Toc417482130)

[6.1.6 debug 14](#_Toc417482131)

[6.1.7 error 14](#_Toc417482132)

[6.1.8 \_exit 14](#_Toc417482133)

[6.1.9 set\_header 15](#_Toc417482134)

[6.1.10 dom 15](#_Toc417482135)

[6.1.11 site\_url 15](#_Toc417482136)

[6.1.12 cur\_url 15](#_Toc417482137)

[6.1.13 curl 15](#_Toc417482138)

[6.1.14 is\_ajax 15](#_Toc417482139)

[6.1.15 is\_post 15](#_Toc417482140)

[6.1.16 is\_get 16](#_Toc417482141)

[6.1.17 is\_delete 16](#_Toc417482142)

[6.1.18 mime 16](#_Toc417482143)

[6.1.19 redirect 16](#_Toc417482144)

[6.1.20 absint 16](#_Toc417482145)

[6.1.21 longint 16](#_Toc417482146)

[6.1.22 move\_file 16](#_Toc417482147)

[6.1.23 referer 16](#_Toc417482148)

[6.1.24 msubstr 17](#_Toc417482149)

[6.1.25 is\_mail 17](#_Toc417482150)

[6.1.26 is\_url 17](#_Toc417482151)

[6.1.27 is\_qq 17](#_Toc417482152)

[6.1.28 is\_mobile 17](#_Toc417482153)

[6.1.29 is\_phone 17](#_Toc417482154)

[6.1.30 fuzzy\_email 17](#_Toc417482155)

[6.1.31 fuzzy\_mail 18](#_Toc417482156)

[6.1.32 send\_mail 18](#_Toc417482157)

[6.1.33 trun\_x 18](#_Toc417482158)

[6.1.34 trun\_y 18](#_Toc417482159)

[6.1.35 is\_id\_card 18](#_Toc417482160)

[6.1.36 is\_datetime 18](#_Toc417482161)

[6.1.37 is\_date 18](#_Toc417482162)

[6.1.38 is\_time 18](#_Toc417482163)

[6.1.39 is\_ip 19](#_Toc417482164)

[6.1.40 string2array 19](#_Toc417482165)

[6.1.41 get\_client\_ip 19](#_Toc417482166)

[6.1.42 json\_array 19](#_Toc417482167)

[6.1.43 get\_object\_vars\_deep 19](#_Toc417482168)

[6.1.44 is\_absint 19](#_Toc417482169)

[6.1.45 price 19](#_Toc417482170)

[6.1.46 add\_task 20](#_Toc417482171)

[6.1.47 U 20](#_Toc417482172)

[6.1.48 val 20](#_Toc417482173)

[6.1.49 check\_not\_empty 20](#_Toc417482174)

[6.1.50 http404 21](#_Toc417482175)

[6.1.51 http\_status 21](#_Toc417482176)

[6.2 Lib的一些工具类 21](#_Toc417482177)

[6.2.1 Lib\Ip\Ip 21](#_Toc417482178)

[6.2.2 Lib\VerifyCode\VerifyCode 21](#_Toc417482179)

[6.2.3 Lib\Page 22](#_Toc417482180)

[6.2.4 Lib\Pinyin 22](#_Toc417482181)

[6.2.5 Lib\Validator 22](#_Toc417482182)

[6.2.6 Lib\Common 22](#_Toc417482183)

[6.2.7 Lib\Csv 24](#_Toc417482184)

[6.2.8 Lib\Encrypt 25](#_Toc417482185)

[6.2.9 Lib\Encrypt2 25](#_Toc417482186)

[6.2.10 Lib\Kvdb 25](#_Toc417482187)

[6.2.11 Lib\Loader 25](#_Toc417482188)

[6.2.12 Lib\Loger 25](#_Toc417482189)

[6.2.13 Lib\Mimes 26](#_Toc417482190)

[6.2.14 Lib\Mongodb 26](#_Toc417482191)

[6.2.15 Lib\Sms 27](#_Toc417482192)

[6.2.16 Lib\Sphinx 27](#_Toc417482193)

[6.2.17 Lib\Widget 27](#_Toc417482194)

# 引言

Cxsv框架是用swoole扩展编写的PHP应用服务MVC框架，运行后以进程形式长驻后台，通过NGINX代理，接受http请求，并调用相应的控制器处理请求。

# 框架开发规范

全框架实行命名空间约束，每定义一个类，都要有命名空间。

## 控制器Action编写规范

1. 每个Action的命名空间为Act开头，类名与文件名相同，首字母大写，Act类继承必须继承自Lib\Action类
2. 每个Action只提供 get , post , add , (put , update) , delete , ajax等6种请求方式
3. 对应的控制器内部方法为：doGet , doPost , doAdd , doUpdate , doDelete , doAjax
4. 默认的请求方法为 get ，对应控制器的doGet方法
5. 一个完整的控制器看起来像下面这样子：

<?php

/\*\*

\* Auth Sang

\* Desc 商品详情页

\* Date 2015-02-10

\*/

namespace Act;

use Lib\Action;

class Test extends Action{

public function doGet(){

#do something

}

public function doPost(){

#do something

}

}

你也许注意到在这个示例里面，没有doUpdate、doAdd等方法，是的，当你不需要的时候，完全可以不必写出来，访问一个不存在的方法时，会返回404错误

每个Action有一个可实现的\_init方法，如果实现该方法，则无论何时，控制器总是会首先执行该方法。你可以在该方法里定义一些公共属性，或做一个用户行为判断等

## 控制器与访问路径的关系

如http://www.ABC.com/index 对应的Action为Act/Index.php，并且调用了doGet方法，相当于<http://www.ABC.com/index/get>

[http://www.ABC.com/index/](http://www.abc.com/index/)method：如果method在上述7种方式中，则调用了控制器内部的相应的方法，如果不在上述7种方式中，则代表一个路径，相应地，调用了Act/Index/ method/get

## 模型Model编写规范

每个Model的命名空间为App\Mod开头，类名与文件名相同，首字母大写，且必须继承自Lib\Model类

每个Mod类看起来像下面这样子：

<?php

/\*\*

\* Auth Sang

\* Desc 商品模型类

\* Date 2015-02-10

\*/

namespace App\Mod;

use Lib\Model;

class Goods extends Model{

public function getGoodsById($id){

if(empty($id) || !is\_numeric($id)){

return false;

}

return $this->where("id={$id}")->fetch();

}

}

你可以在这里定义一些方法，以供控制器访问。当然，你也可以不定义任何方法，只定义一个类的主体，在应用里面，你仍然可以通过调用Lib\Model里面提供的方法进行数据的增删改查。但是一般不建议这么做，并且在可能的情况下，会强制不要这么做。把数据的获取方式定义在Mod类里比较可靠和安全，可维护性也要好很多。

# 目录结构

## 应用根目录

应用的根目录为静态常量DOCUMENT\_ROOT

## 应用目录结构

### 不分前后台的情况下的目录结构

Webroot

Act (控制器存放目录)

├─ Index.php

├─ Test.php

Mod (模块存放目录)

├─ Goods.php

Lib (应用公共库目录)

├─ Net.php

Plugin (插件目录)

├─ onResponse

├─ onResponse.php

Widget (挂件目录)

├─ Ad

├─ Ad.php

Tpl (模板文件目录)

├─ default (模板样式)

├─ index (对应一个Action，全小写)

├─ index.php (模板文，全小写)

├─ form.php

Task(异步任务脚本目录)

|-- Task.php

Cron(定时器目录)

|-- Cron.php

### 分前后台的情况下应用的目录结构

Webroot

admin

├─ Act (Action存放目录)

├─ Index.php

├─ Test.php

├─ Tpl (模板文件目录)

├─ default (模板样式)

├─ index (对应一个Action，全小写)

├─ index.php (模板文，全小写)

├─ form.php

front

├─ Act (Action存放目录)

├─ Index.php

├─ Test.php

├─ Tpl (模板文件目录)

├─ default (模板样式)

├─ index (对应一个Action，全小写)

├─ form.php

Mod (应用模块存放目录)

├─ Goods.php

Lib (应用公共库目录)

├─ Net.php

Plugin (插件目录)

├─ onResponse

├─ onResponse.php

Widget (挂件目录)

├─ Ad

├─ Ad.php

Task(异步任务脚本目录)

|-- Task.php

Cron(定时器目录)

|-- Cron.php

## 命名空间对应目录

### Lib

全局公共类库，对应框架根目录/lib

### Mod

全局公共model，对应框架根目录/mod

### App

DOCUMENT\_ROOT

### Act

DOCUMENT\_ROOT/Act

### Widget

DOCUMENT\_ROOT/Widget

### Task

DOCUMENT\_ROOT/Task

# 注意事项

由于常驻后台，并且没有用到PHP-FPM或APACHE，有一些操作需要注意

## 关于exit和die退出语句的使用

上式上线应用不能用exit或die退出脚本，否则会导致进程重启，取之代之的是\_exit($msg)函数，并且调用了\_exit()之后，之前的所有输出都会被清空。如果需要在\_exit()之后打印内容，应将要打印的内容当做\_exit()的参数传入

## 关于header函数

Header()函数将失败，取而代之的是set\_header($header\_key,$header\_value)函数

## 关于static关键字

在函数内使用static声时变量要注意，由于常驻后台（常驻内存）的特性，会导致声明static变量后，该变量也常驻内存，如果需要对该变量进行相加或数组赋值操作，将会造成该变量在内存中一直增长。解决方法是用类，定义一个类，将static变量用类的属性代替。在类中对该属性进行初始化。每次使用时，NEW这个类

## 关于cookie

cookie访问方式有两种，一种是原生的$\_COOKIE[‘key’]方式，一种是使用框架提供的cookie()函数

### cookie函数的使用

cookie(‘key’)

获取key的cookie内容

cookie (‘key’,’val’)

设置key为val的cookie

cookie (‘cookie’,’val’,1800)

设置key为val的cookie，过期时间单位为1800秒

cookie (‘key’,null)

删除key对应的cookie

## 关于session

session的访问方式有两种，一种是原生的$\_SESSION[‘key’]，一种是使用框架提供的session()函数

### session函数的使用

Session(‘@id’)

返回session\_id

session(‘key’)

获取key对应的session内容

session (‘key’,’val’)

设置key为val的session

session (‘key’,’val’,1800)

设置key为val的session，过期时间为1800秒，如不传入此参数，则过期时间受应用配置控制

session (‘key’,null)

删除key对应的session内容

session (null)

删除所有session内容

## 关于储存storage

框架为应用提供了统一储存，通过使用storage函数，可方便地上传，获取，删除储存文件

### storage函数的使用

storage(‘@/path/to’,[‘path’=>’path’])

保存一个本地文件到存储系统

参数解释：

第一个参数为一个本地文件，注意前面加@，使用绝对路径

第二个参数为一个关联数组，数组每个项的含义如下：

path：将文件保存在储存系统中的路径，文件名由系统自动生成

返回值：

成功返回true，失败返回false

storage(‘strings’,[‘path’=>’path/to’])

保存内容到文件文件系统的

参数解释：

第一个参数为要保存的内容

第二个参数为一个关联数组，数组项含义如下：

path：将内容保存在储存系统中的路径，文件名由系统自动生成

返回true或false

storage(‘path/to’)

从存储系统中获取一个文件

参数1为文件在储存系统中的相对路径

storage(‘path/to’,null)

从存储系统中删除一个文件

参数1为文件在储存系统中的相对路径

## 关于配置项config

应用配置分为系统级配置和应用级配置，系统级配置在框架定义配置文件，一般不可修改。

系统级配置访问方式为：

function(){

global $php;

$val = $php->c(‘key’)

}

需要函数/方法内部global $php这个变量

应用级配置访问方式为调用框架提供的config函数

### config函数的使用

config(‘键’)

获取一个配置项

config(‘键’,’值’)

设置一个配置项

config(‘键’,null)

删除一个配置项

## 如何获取一个模型 model

要实例化一个模型类，只需通过框架提供的model函数即可

### model函数的使用

model(‘#类名’)

调用公共mod

你也许注意到参数前面的#号，当参数前面加#时，表示调用框架提供的公共Mod

model(‘类名’)

调用应用根目录/Mod下的模型类

model(‘类名’,’数据表前缀’)

调用带数据表前缀的mod，例如 model(‘admin’,’adm\_’);将调用应用目录/Mod下的Admin类，并操作 adm\_admin的表

类名同时也是小写的数据表名

## 关于缓存cache

框架提供cache函数来操作缓存，基于memcache

### cache函数的使用

cache(“缓存名”)

获取对应的cache内容

cache(“缓存名”,”设置内容”)

设置对应缓存名的内容

cache(“缓存名”,”设置内容”,过期时间)

设置对应缓存名的内容，过期时间单位为秒

cache(“缓存名”,null)

删除对应缓存的内容

# 开发示例

## 数据的增删改查

$model=model(“类名”);

-----增-------

$result = $model->insert(array('name'=>'value','name2'=>'value2'))

$result=$model->multiInsert(array(array('name'=>'value','name2'=>'value2')),array('name'=>'value','name2'=>'value2')))

同时插入多条

成功返回插入的ID，失败返回FALSE

-----删-------

$result = $model->where($where)->delete();

成功返回受影响行数，失败返回FALSE

-----改-------

$result = $model->where($where)->update(array('name'=>'value','name2'=>'value2'));

成功返回true或受影响行数，失败返回FALSE

-----查-------

返回数据集（多条记录）

$data=$model->where($where)->limit(offset,resultNumrows)->order(“id desc”)->select();

单条记录

$data=$model->where($where)->limit(offset,resultNumrows)->order(“id desc”)->fetch();

注：where条件为一个sql字符串，如：id=1 或 id>1 and id<1、id!=1 、id in(1,2,3) and title like '%test%'等

# 附录

## 常用的一些公共函数

### get

获取$\_GET变量的值：

get('变量名','默认值','处理函数，多个之间用逗号分隔');

如：get('foo',0,'intval,abs');处理函数的顺序是从左到右。

### post

获取$\_POST变量的值：

post('变量名','默认值','处理函数，多个之间用逗号分隔');

如：post('foo',0,'intval,abs');处理函数的顺序是从左到右。

对用户的输入要严格过滤。程序需要的变量是什么类型，就要过滤成什么类型。

### request

获取$\_GET或$\_POST的值

request($key,$default,$funs);

如：request(‘foo’,0,’intval,abs’);

### add\_action

添加一个钩子，参数：类型，如：onrequest；处理函数；权重

add\_action($type,$handler,$weight);

### apply\_action

执行钩子，参数：类型

apply\_action($type);

### debug

输出调试信息并结束脚本运行，传入一个或多个参数;可以是任意数据类型

debug();

### error

抛出一个错误，参数：错误提示，代码

error($msg,$code);

### \_exit

退出脚本，不能用exit 要用 \_exit()

\_exit(退出提示);

### set\_header

设置响应头;用以代替原header函数；

set\_header($key,$val)

如：set\_header(‘Content-Length’,1024);

### dom

输出自定义二级域名的网址，参数：二级域名，如：gz

dom($dom\_name);

例：dom(‘gz’)

输出：gz.domain.com

dom()

输出：domain.com

### site\_url

获取带http:// 的网站根网址

site\_url()

### cur\_url

获取当前访问的链接

cur\_url()

### curl

获取网址的内容-采集，参数：网址，方法（Get或Post），post表单字段，

curl($url,$method,$postfields)

### is\_ajax

判断是否ajax请求

is\_ajax()

### is\_post

判断是否Post请求

is\_post()

### is\_get

判断是否get请求

is\_get();

### is\_delete

判断是否DELETE请求

Is\_delete();

### mime

获取文件资源的mime类型，参数：文件绝对路径

get\_mime($path);

### redirect

URL跳转，代替header(‘location:url’)

redirect($url)

### absint

数参数转换为正整数，参数：任意数字和字符串。非数字返回0

absint($num)

### longint

处理长整数，解决系统对数字的最大限制问题

longint($num)

### move\_file

移动文件到指定路径，参数：源文件路径，目录路径

move\_file($source,$dest);

### referer

获取来路地址，参数$url不为空时，为保存$url到session，下次获取时会返回该url

referrer($url)

### msubstr

截取字符串，支持中文和utf8编码，参数：要截取的字符串，开始位置，长度，编码（默认为utf8），超出字符用什么代替

msubstr($str,$start,$length,$charset,$suffix);

### is\_mail

判断是不是有效的email地址

is\_mail($email)

### is\_url

判断是否有效的url

is\_url($url)

### is\_qq

判断是否有效的QQ号码

is\_qq($qq);

### is\_mobile

判断是否有效的手机号码

is\_mobile($mobile)

### is\_phone

判断是否有效的电话号码

is\_phone($phone);

### fuzzy\_email

字符串模糊化，参数：要模糊化的字符串，开始位置，长度

fuzzy($str,$start,$length)

### fuzzy\_mail

邮箱地址模糊化

fuzzy\_email($email)

### send\_mail

发送邮件，参数：接收方邮箱，标题，正文，附件（以数组形式）

send\_mail($to,$subject,$body,$attachment);

### trun\_x

图片x轴翻转，参数：图片绝对路径，图片类型（如：jpeg,gif,png）

trun\_x($img\_file\_path,$type)

### trun\_y

图片y轴翻转，参数：图片绝对路径，图片类型（如：jpeg,gif,png）

trun\_y($img\_file\_path,$type);

### is\_id\_card

判断是否有效的身份证号码，参数：身份证号码

is\_id\_card($id\_card)

### is\_datetime

判断是否有效的时间格式，参数：0000-00-00 00:00:00 格式的时间

is\_datetime($datetime)

### is\_date

判断是否日期

is\_date($date)

### is\_time

判断是否时间

is\_time($time)

### is\_ip

判断是否IP地址

is\_ip($ip);

### string2array

以逗号分隔的字符串转换成数组

string2array($str)

### get\_client\_ip

获取客户端IP地址

get\_client\_ip();

### json\_array

JSON数据转换成数组

json\_array($json)

### get\_object\_vars\_deep

递归将对象变成数组

get\_object\_vars\_deep($object);

### is\_absint

/\*\*

\* 检查是否正整数

\* @param numeric $num

\* @return bool

\*/

is\_absint($num);

### price

/\*\*

\* 将数字转换成2位数的金钱

\* @param numeric $num

\* @return float

\*/

price($num);

### add\_task

/\*\*

\* 添加一个异步任务

\* @param string $task\_name 任务名称

\* @param arrau $task\_data 任务数据

\* @return bool

\*/

add\_task($task\_name,$task\_data);

### U

/\*\*

\* 返回带域名的链接

\* @param string $path

\* @return string

\*/

U($path);

### val

/\*\*

\* 根据数组或对象的下标取值，不存在则返回默认

\* @param mix $obj

\* @return string

\*/

val($obj,$key,$default='');

### check\_not\_empty

/\*\*

\* 检查所有入参是否不为空值

\* @param mix 传入任意要检测的变量

\* @return bool

\*/

check\_not\_empty();

### http404

/\*\*

\* 发送404状态码

\* @param string $title

\* @param string $content

\* @return void

\*/

http404();

### http\_status

/\*\*

\* 发送自定义状态码

\* @param string $title

\* @param string $content

\* @return void

\*/

http\_status($title,$content,$code=502);

## Lib的一些工具类

调用方法为：

引用命名空间方式：use Lib\ClassName; $class = new ClassName();

需要namespace 下面加一行：use Lib\ClassName;

直接使用方式：$class = new \Lib\ClassName();

### Lib\Ip\Ip

判断IP地址所在地的类库

use Lib\Ip\Ip;

$ipquery = new Ip();

$location = $ipquery->getLocation($client\_ip);

返回：Array ( [ip] => 219.136.32.110 [beginip] => 219.136.32.0 [endip] => 219.136.34.255 [country] => 广东省广州市海珠区 [area] => 电信ADSL )

### Lib\VerifyCode\VerifyCode

输出验证码

use Lib\VerifyCode\VerifyCode

VerifyCode::show($ch); //当$ch为true时，输出带中文的验证码

### Lib\Page

输出分页代码

use Lib\Page;

$page = new Page($count,$size); //$count:数据总数，$size:每页数据量

$page->offset //数据偏移量

$page->size //每页数据量

$page->show() //输出分页代码

### Lib\Pinyin

中文转拼音

use Lib\Pinyin;

$pinyin = Pinyin::Pinyin($str,$charset) //$str:中文字符串,$charset:字符串编码，默认为utf8

### Lib\Validator

数据验证类

use Lib\Validator;

$validator = new Validator($data); //$data:要验证的数据

$validator->setRules([ //设置验证规则

'field' => ['required'=>true,'validate'=>'trim'],

])->setMessages([ //设置验证失败时提示语

'field' => ['required'=>'字段不能为空','validate'=>'字段数据不合法'],

]);

$data = $validator->validate();

if(!$data){

$this->error = $validator->getError();

return false;

}

### Lib\Common

Common类，封装一些常用方法

use Lib\Common

#### Common::getActionUrl()

Common::getActionUrl() //获取当前访问的request\_path 如：http://www.dom.com/path/to　　返回 /path/to

#### Common::getIp()

Common::getIp() 获取客户端IP地址

#### Common::getDateTime($time)

将时间戳格式化为：2014-01-01 12:00:00

#### Common::getDate($time)

将时间戳格式化为：2014-01-01

#### Common::getRandomIp()

随机输出一个ＩＰ地址

#### Common::getRandString($length=6)

产生一个$length长度的随机数字

#### Common::filterText($str)

过滤文本里的单引号

#### Common::renderJsConfirm($class,$confirm='确定要这样做吗')

在页面中添加某个元素的点击提示，基于bootstap

#### Common::day2year($day)

如果$day正好是12的倍数，则输出 ($day/12)年，否则输出$day个月

#### Common::renderJsRegion($level=3)

生成$level级联动的地区选择框

#### Common::genOrderSn()

产生一个唯一的订单号

#### Common::getBankList()

获得银行代码列表，如：["ABC"=>"中国农业银行",..........]

#### Common::genPassword($password,$salt="")

产生密码 双重MD5 加扰码，返回[加密后的密码,$salt]。如果传入$salt，则$salt原样返回

#### Common::getPayPassword()

产生一个支付密码

### Lib\Csv

生成一个csv文件并可下载

use Lib\Csv;

$csv = new $Csv();

$csv->setFileName($filename) //设置生成后的文件名

$csv->put($data) //添加一行数据 $data为一个顺序索引一维数组

$csv->create() //生成csv文件

$csv->download() //下载生成的文件

### Lib\Encrypt

AES加密解密类

use Lib\Encrypt;

$encode = Encrypt::encode($text,$key='') //加密内容，$key为密钥，如果为空，则用内置的

$decode = Encrypt::decrypt($encode,$key='') //解密加密的内容，$key为密钥，如果为空，则用内置的，注意此key必须和加密时的key一致，否则无法解密

### Lib\Encrypt2

AES加密解密类，提供与java一致的加密算法

use Lib\Encrypt2;

$encode = Encrypt2::encode($text,$key='') //加密内容，$key为密钥，如果为空，则用内置的

$decode = Encrypt2::decrypt($encode,$key='') //解密加密的内容，$key为密钥，如果为空，则用内置的，注意此key必须和加密时的key一致，否则无法解密

### Lib\Kvdb

基于mysql和memcache的持久化缓存类

use Lib\Kvdb;

$kvdb = new Kvdb($table\_name); //实例化 $table\_name为数据库对应的表名

$kvdb->get($name) //获取一个缓存值

$kvdb->set($name,$value) //设置一个缓存值

$kvdb->delete($name) //删除一个缓存

config()函数基于此类构造

### Lib\Loader

命名实间加载类

use Lib\Loader;

Loader::setRootNS($namespace,$path); //设置命名空间对应的项目路径

### Lib\Loger

日志输出类

use Lib\Loger;

$loger = new Loger($prefix=''); //实例化 $prefix为日志文件前缀

$loger->info($str); //写日志，级别为 info

$loger->error($str) //写日志，级别为error

$loger->自定义($str) //写日志，自定义级别

### Lib\Mimes

对一些常见mime类型的定义和获取

use Lib\Mimes;

Mimes::types() //返回如：['gif'=>'image/gif']

Mimes::mimes() //返回如：['image/gif' => 'gif']

### Lib\Mongodb

对mongodb的操作

已用mongodb($table\_name)封装

使用方法：

$mdb = mongodb('table\_name');

//增加数据，单条插入

$mdb->insert($data,$opts=[]); //$data为关联数组，$opts参考mongodb文档

//增加数据，多条插入

$mdb->multiInsert($data,$opts=[]) //$data为二维索引关联数组，如：[['a'=>'b'],['b'=>'c']]

//删除数据

$mdb->where($where)->delete(); // $where可以是sql语句的where条件，也可以是mongodb的数组条件

//删除表

$mdb->drop();

//修改数据

$mdb->where($where)->update($new\_data=[]); // $where可以是sql语句的where条件，也可以是mongodb的数组条件

//查询 返回数据集

$mdb->where($where)->limit($firstRow,$pageSize)->fields($fields="\*")->order("id desc")->select();

//查询 返回单条数据

$mdb->where($where)->limit($firstRow,$pageSize)->fields($fields="\*")->order("id desc")->fetch();

//执行mongodb命令

$mdb->command($command);

//返回错误

$mdb->getError(); //当操作发生错误时，返回错误

### Lib\Sms

短信发送类

use Lib\Sms;

$sms = Sms::getInstance(); //单例模式，获得实例

$sms->setGateway(); //设置短信接口网关地址

$sms->params = $value; //设置短信接口系统参数，如用户名，密码之类

$sms->setParams([key=>val]) //设置业务参数，如接收手机，发送内容等

$ret = $sms->send(); //发送 ，返回接口的状态码

### Lib\Sphinx

Sphinx插索引掣接口集成

use Lib\Sphinx;

使用方法参考Sphinx文档

### Lib\Widget

挂件类

已用Act->widget($widget\_name,$data=[]) 封装

在Action里这样调用：

$this->widget('show\_ad',['size'=>4,'ad\_pos'=>121212]); //$widget\_name 为挂件名，对应 App/Widget/WidgetName/WidgetName.php，$data 为向挂件传递数据

# 异步任务脚本的写法示例：

/\*\*

\* Auth Sang

\* Desc 商品PV

\* Date 2015-02-12

\*/

namespace Task;

class Goods{

public static function run($data){

global $php;

try{

$data = $data;

$mod = model('Goods');

//PV+1

$mod->pv($data['id'],1);

}catch(\Exception $e){

$php->log($e->getMessage.' | '.$e->getFile());

}

}

}

# 定时脚本写法示例：

/\*\*

\* @Auth Sang

\* @Desc test

\* 时间格式，秒 分 时 日 月 周,用法与linux的crontab相同

\* 一定要定义公有变量 $time，否则该定时任务不工作

\* @Date 2015-03-13

\*/

namespace Cron;

class Test{

public $time = '\*/3 \* \* \* \* \*';

private $start\_time;

public function \_\_construct($start\_time){

$this->start\_time = $start\_time;

}

public function run(){

#do something

}

}