KYPHP 框架手册

版本 v2.1 作者：张亮

版本历史：

版本	说明
V1.0	第一版 kyphp，支持 kysmarty（kyphp 控件）,支持纯数据层 M,简单的 mvc,有路由分发，多模板，多语言
V1.1	支持文件自定义，增强 datalist 控件,优化缓存
V1.2	增强 model 结构,修正 url 静态模式,增加 memcache 类,增加 load 方法
V1.2.8	从 1.2 版本开源，增加函数，如 is_utf8，curl_file_get_contents 等，加入了 mssql 的支持，完美支持 access 以及 sqlserver ，加入了 session 类,cookie，分页类等
V2.0	此版不仅兼容了历史版本,而且重定义了 mvc 结构，并支持自定义类，自动加载类，视图类增强支持 render 和 display，kysmarty 增强支持部分 smarty 功能，文件结构自定义性更强，支持多级目录，支持多 app 方便拆分，支持多数据库驱动，多 cache 驱动，支持同时使用多数据库，多 cache 并增强多语言库的使用,从此 kyphp 不仅能快速搭建简单应用，还支持大型项目
V2.1	在 v2.0 基础上支持 smarty，支持 mysqli bind 用法，安全上做了很大处理，$db->where()建议用数组，debug 模式增加了每个类的执行时间，示例增强

QQ:974005652 群：105718680 Email:zhx626@126.com

网站：http://www.ky53.net

官方微博：http://weibo.com/u/3894968041

作者：张亮(老顽童)QQ:974005652 群：105718680 网站：http://www.ky53.net

KYPHP 框架手册

目录

简介

KYPHP 是经过两年的开发，终于从原来的 v1.0 版本到成熟版本 v2.0,成熟版更强大，KYPHP2.0 不仅支持以前所有的功能，还支持多数据库，多语言，多模版，多 app,多缓存，多编码格式，模板布局，自定义类，自动加载公共类库。其扩展性和可用性堪比 yii、zend 等框架，并拥有高效简洁的语法，使你能在大项目中游刃有余。

一、入口文件及配置

1.入口文件

```
<?php
define('APP_PATH',dirname(__FILE__));//当前项目路径
define('KYPHP_PATH',APP_PATH.'/../kyphp/');//KYPHP 框架路径

require KYPHP_PATH."kyphp.php";

KYPHP::Run();

/*
//如果要自定义 config 路径，需要定义__CONFIG__
define('__CONFIG__',APP_PATH.'/inc/config.php');
$config=require(__CONFIG__);
KYPHP::Run($config);
*/


?>
```

2.配置 config

```
<?php
if (!defined('KYPHP_PATH'))   exit('未定义 KY_PATH');
$array = array(
    'WEBURL'=>'http://localhost/kyphpfrm2.0/blog',//网站 url
    'SSLURL'=>'https://localhost/kyphpfrm2.0/blog',//如果有 ssl，https 则配置此处
    'URL_ROUTER_ON' => true,
    'DEFAULT_MODULE' =>'home/index',//默认路由
    'PATH_KEY'=>5, // URL 类型,兼容模式请设置为 3     //2 是伪静态  1 是 pathinfo 4 为静态,5 为自定义 url
    'URL_MODULE'=>'public/url',//PATH_KEY 为 5 时必须指定,url 类的路由
```

```
'DB_CHARSET'=>'utf8',
'DB_HOST'=>'localhost',
'DB_NAME'=>'kyfrm_utf8',
'DB_USER'=>'test',
'DB_PWD'=>'123456',
'DB_PREFIX'=>'ky_',//数据库前缀
'DB_TYPE'=>'mysql',数据库类型，可选 mysql,pdo_mysql,mysqli,mmsql,postgre 等
'_db'=>array(
    'mysqli'=>array('DB_CHARSET'=>'utf8','DB_HOST'=>'localhost',
'DB_NAME'=>'kyfrm_utf8',  'DB_USER'=>'test',  'DB_PWD'=>'123456','DB_PREFIX'=>'ky_',
'DB_TYPE'=>'mysqli'),//$this->_db['db1']
    'pdo'=>array('DB_HOST'=>'localhost',  'DB_NAME'=>'kyfrm',  'DB_USER'=>'test',
'DB_PWD'=>'123456','DB_PREFIX'=>'ky_',  'DB_TYPE'=>'pdo_mysql'),// 多数据库设置，
$this->_db[$key]键值$this->_db[1]

),
'cache'=>array('driver'=>'file',//支持 redis,file,memcache
                'host'=>'127.0.0.1',
                'port'=>'11211',

),
'_cache'=>array(
    //'memcache'=>array('driver'=>'memcache','host'=>'127.0.0.1','port'=>'11211'),// 多缓
存使用
    'file'=>array('driver'=>'file'),

),
'app_dir'=>'app',//项目文件夹 支持多 app，可设置为数组 array('app','app2','app3');默认
第一个 app,可在 url 中用&app=app2 来读取其它 app，如果 rewrite 请选用模式 5 自定义 rewrite
//'DEFAULT_M_PATH' =>'m',//指定生成 M 文件路径 空为'model'
//'DEFAULT_C_PATH' =>'c',//指定生成 LIB 文件路径，空为'controller'
//'DEFAULT_V_PATH' =>'v',//指定生成 TPL 文件路径，空为'tpl'
//'DEFAULT_L_PATH' =>'l',//指定生成 TPL 文件路径，空为'language'
//'DEFAULT_CMD_PATH'=>'common',//指定生成 TPL 文件路径，空为'common'
'error_filename'=>'log.txt',//日志文件名
'error_display'=>'on',//输出错误
'error_log'=>'on',//记录错误
'debug'=>'on',//开启 debug trace
'DIR_LOGS'=>'log',//错误日志目录
'templete_ext'=>'.html',//模板扩展名
'code'=>'zh-cn',//默认 zh-cn 当前语言
'   DIR_CACHE'=>'cache',//缓存目录
'templete_mode'=>'smarty',//smarty,kysmarty 默认为 kysmarty
'smarty_library'=>'../Smarty-3.1.19/libs',//smarty libs 的路径
```

```
    'smarty_left_delimiter'=>'{',//smarty left_delimiter
    'smarty_right_delimiter'=>';}',//smarty right_delimiter 兼容 kysmarty
    //'smarty_path'=>'', smarty path
);
return $array;

?>
```

3.目录结构及文件名和类名定义

V1 文件结构

/cmd->cmd.php

/inc->config.php

/lib->public.class.php index.class.php

/tpl->default/

/temp->cahe/

/index.php 项目入口文件

Controller 的文件命名是 classname.class.php，类的命名是 classnameAction

可以继承公共类 publicAction

Model 的文件命名是类名.class.php 类的命名,类的命名是 classnameModel，可用 load 方法调用$this->load('classname'); $this->model_ classname'->method();

V2 文件结构

/app/controller/功能文件夹/index.php info.php
/app/model/功能文件夹/data.php
/app/language/语言/功能文件夹/data.php
/app/tpl/default/功能文件夹/index.html 或.tpl
/app1/同 app mvc 结构
/app2/同 app mvc 结构
/inc/config.php
/log
/public 可由开发者定义，存放 css ,image,js
/ custom 自定义类，可放外部加载库
/cache
/common/cmd.php 同/cmd，存放函数文件 cmd.php
Controller 类名定义：ControllerHomeIndex Controller 文件夹名文件名，可继承 extends ControllerPublicPublic
Model 类定义 ModelBlogData Model 文件夹名文件名
V2 兼容 v1

二、常量定义

1.路径类

KYPHP_PATH/KY_PATH 是框架的路径,KY_PATH 已废弃
__CONFIG__ 是系统 config.php 的路径，包含 config.php，当然你可以命成其它名字
APP_PATH 是当前项目的路径
DIR_CACHE 缓存目录，默认为项目下 cache,需要在 config.php 设置，对应 config 里 DIR_CACHE
DIR_LOGS 日志目录，记录错误或用户自定 log，需要在 config 里设置,对应 config 里 DIR_LOGS
DEFAULT_M_PATH 项目 model 层路径
DEFAULT_C_PATH 项目 controller 层路径，在 v1 版本中同 DEFAULT_LIB_PATH
DEFAULT_V_PATH 项目 view 层路径，在 v1 版本中同 DEFAULT_TPL_PATH
DEFAULT_L_PATH 项目 language 层路径
DEFAULT_CMD_PATH 默认 common 的路径，即 cmd.php （function）的路径

2.设置类

__CHARSET__字符编码设置 utf-8,gbk,gb2312
__WEBURL__网站 url
DEFAULT_MODULE 默认的路由，默认为 index/index
__URL__当前路由的 url
__APP__当前项目的 url
DEFAULT_TPL 是默认的 tpl 文件夹,默认为 default,在 config 里是 DEFAULT_TPL

三、模板使用

1、使用 include <include file="public:top"> 可调用 public 文件夹下的 top.html

2、使用渲染,

```
$this->data['header']=$this->view('public/header'); 可调 public 路由的 header
$this->data['footer']=$this->view('public/footer');
$this->data['right']=$this->view('public/right');//调
```

3、向模板传值

$this->data[] 等同于$this->assign($key,$value);

```
$this->lable(“lablename”,”hello world!”)

$this->volist(“list”,array)

$this-> volist (“list”,array,”list_1”,array) 中的 list_1

$this->datalist(“list”,array)
```

KYPHP 标签

标签类：

3.1.<lable>

语法：<lable name=”lablename”>表示一个 label，name 的值，对应该文件 actionclass 的$this->lable(“lablename”,”hello world!”)

为了方便可以将该控件简化为$lable[lablename]

3.2.<volist>

语法：<volist name=”list” id=”vo”>**</volist>表示一个循环列表，对应该文件 actionclass 的$this->volist(“list”,array)

**中用{$vo.title}或{$title}直接表示值

3.3.<list>:用法同<volist>用在其嵌套中对应的值是

$this->volist(“list”,array,”list_1”,array) 中的 list_1

3.4.<datalist>

语法：<datalist name=”list” id=”" row=”" field=”字段|名称:js%css” manage=”URL|显示名:js%css” edit=”true” title=”true” page=”true” table=”table 属性” list=”table 或 li”>

对应文件 actionclass 的$this->datalist(“list”,array)

四、基类的使用

基类包括

4.1 Db 类

new DB($driver, $hostname, $username, $password, $database,$dbpre='',$chrset='utf8'),
可以在 KYPHP 控制器,示图，数据层用$this->db,或$this->_db[],方法有
$this->db->query($sql);执行 sql 并获取$query->row,$query->rows 等
$this->db->exec($sql);执行 sql;通常是 insert 和 update,delete
Function tablename($name)可以获得带有前缀的表名
Function getLastId()获得插入后的 id, $this->db-> getLastId();
function select($fieldstr='*')及 findAll()，可获得查询 select，需要指定表$db->table,可用 D()或 M 方法获得
function where($sting) 指定查询的 sql 条件
function find()得到单条数据
function total()得到总数
function order($string)进行排序
function limit($limit) 取选定的个数
function field($field) 指定查询中的字段
function add($data=array()) 添加数据，$data 为要添中的数据,array($key=>$value, $key=>$value)
public function addAll($data=array()) 添加所有数据，即可以是表单中的，完成自动填表，也可以是指定的 $data
public function save($data=array())保存数据，即 update
public function delete()删除数据
public function saveAll($data=array()) 自动完成填表，update $data=array($key=>$value, $key=>$value)
public function escape($value) 转义字符串

4.2 Cache 类

new cache(array('driver'=>'memcache', 'host'=>'127.0.0.1','port'=>'11211'))
可以在 KYPHP 控制器,示图，数据层用 $this->cache, 或 $this->_cache[], 方法有
$this->cache->get($key),
$this->cache->set($key,$value,$time_expire),
$this->cache->delete($key);

4.3 language 类，可用$this->language->get()得到语言包

4.4 log 类，new log('文件名'),可以用$this->log->wirte($str)输出到系统日志

4.5 image 类，提供水印，压缩等方法，可以用$this->image 方法

function save($file, $quality = 100)保存图像
function resize($width = 0, $height = 0) 缩放图片
function watermark($file, $position = 'bottomright') 添加水印，位置有 topleft，topright，bottomleft，bottomright，center

4.6 pagination 类

用法

```
$pagination = new Pagination();
$pagination->total = $total;
$pagination->page = $page;
$pagination->limit = $count;
$pagination->text = "从 {start} 起到 {end} 页 总 {total}个 总 ({pages} 页)";
$pagination->url = $this->url->link('home/index', '&page={page}');
$this->data['pagination'] = $pagination->render();//获取页码显示
```

4.7config 类，$this->config->get($key)可获得系统设置$this->config->set($key,$value)添加系统设置，

$this->config->get('error')可获得系统当前错误

4.8 cookie 类

Cookie::set($name,$value,$time=0,$path='/',$host='',$secure=0,$httponly =0) 设置 cookie
Cookie::get($key),获得 cookie
Cookie::remove($key);删除 cookie

4.9 json 类 Json:encode()和 Json:decode() 如系统没有 json_encode 函数，可以用 json 类

4.10 url 类 url 路由管理类

public function addRewrite($rewrite) 添加自定义 rewrite 类,可以有很多个
public function link($route='', $args = '', $connection = 'NONSSL') 得取当前的 url,用法 $this->url->link('public/home');
关于自定义 rewrite 类的写法
例子：

```
class ControllerPublicUrl extends Controller{
    public function index() {
        // 添加静态到类
        if($this->config->get('PATH_KEY')==5){
            $this->url->addRewrite($this);
        }
```

```
        // 解码
        if (isset($this->request->get['_k_'])) {
            $parts = explode('/', $this->request->get['_k_']);
            if(strstr($this->request->get['_k_'],'---')){
                $this->request->get['k']='info/view';
            }

            if (isset($this->request->get['k'])) {

                return $this->request->get['k'];
            }
        }
    }

    public function rewrite($link) {

        if($this->config->get('PATH_KEY')==5){
            $url_data = parse_url(str_replace('&amp;', '&', $link));

            $url = '';

            $data = array();

            parse_str($url_data['query'], $data);

            if(strstr($link,'info/view')){
                $url='/---';
            }

            if ($url) {
                unset($data['k']);

                $query = '';

                if ($data) {
                    foreach ($data as $key => $value) {
                        $query .= '&' . $key . '=' . $value;
                    }

                    if ($query) {
                        $query = '?' . trim($query, '&');
                    }
                }
```

```
                return $url_data['scheme'] . '://' . $url_data['host'] . (isset($url_data['port']) ?
':' . $url_data['port'] : '') . str_replace('/index.php', '', $url_data['path']) . $url . $query;
            } else {
                return $link;
            }
        } else {
            return $link;
        }
    }
}
```

需要在 config 中设置'URL_MODULE'=>'public/url',

4.11 request 类，可获得 get ,request,post,server 数组$this->request->get[$key]

4.12 response 类，可以进行输出压缩,添加 header 等

public function addHeader($header)添加 header
public function redirect($url) 跳转
function setOutput($out) 设置输出
public function output() 得到压缩后的输出
在系统类用$this->response-> output();

4.13 runtime 类，获得项目执行时间

$this-> runtime->start()开始计时
$this-> runtime->end()结束计时
function spent() 得到执行时间为多少毫秒
$this-> runtime-> spent ()

4.14 session 类 ,$this->session->data[$key], 得到 session, $this->session->data[$key]='test' 对 sessoin 赋值

4.15 templete 类，可以在项目中临时用模板,

Function set($tpl)设置模板例如$templete=new template(),$templete->set('public/home.tpl')，即模板位置下的 public/home.tpl，传入的值可用$templete ->data=array($key=>$value);
Function render();获得设置好的模板内容输出

4.16action 类，系统 action 类，可以再生成一次控制层，需要传入$KYPHP 类

五、数据库管理

包括 mysql,pdo, postgre,mmsql 等
支持的驱动有 mysql,pdo_mysql,mysqli,mmsql, postgre,当然你可以增加其它的驱动，扩展性非常强。

多数据库请配置_db=array();见配置，章节一

六、缓存管理

包括文件缓存，redis,memcache 等
多缓存请在 config 中配置_cache=array(),见配置，章节一

配置里'CACHE_ON'=>false,//是打开页面缓存
'DIR_CACHE'=>'cache',//cache 路径,仅文件 cache
'CACHE_TIME_EXPIRE'=>120,//设置过期时间
'CACHE_CONTENT_WITHTIME'=>'off',//设置缓存文件内容后缀 off 为关,为空是自带后缀，其它字符串将直接显示在页面底部
'CACHE_HASH'=>'on',//是否用 hash 出的目录，如果页面较多建议设置为 on

七、自定义类的使用

在入口目录下，custom 文件目录可以自定义类，可在项目中自动加载，不必 require

八、路由分发模式和自定义 url 路由

1.见 url 类

在 config 中设置
'PATH_KEY'=>5, // URL 类型,兼容模式请设置为 3 //2 是伪静态 1 是 pathinfo 4 为静态,5 为自定义 url

2.rewrite 写法

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^([^?]+)$ index.php?_k_=$1 [L]      # PATH_KEY 5

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond $1 !^(index.php|images|robots\.txt)
RewriteRule ^html/(.*)$ index.php?k=$1 [L]      # PATH_KEY 4
RewriteRule ^index.php\/(.*)$ index.php?k=$1 [L]      # PATH_KEY 1
RewriteCond %{REQUEST_FILENAME} !-f
```

```
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php?k=$1 [L]     # PATH_KEY 2
```

3.自定义路由解析

Class ControllerPublicUrl 位于 app/controller/public (v2.0 版本后)
方法： public function index()//对 url 进行解析
function rewrite($link) ：在 controller 中使用$this->url->link()将使用该方法实现

九、语言包的使用

见 language 类,文件目录可自定
//'DEFAULT_L_PATH' =>'l',//指定生成 TPL 文件路径，空为'language'

十、模板标签详解

见模板使用

十一、高级应用

1、templete 中使用 smarty ,需要 smarty 类库支持
2、拆分 app ，设置 config 中的'app_dir'=>'app',/
项目文件夹 支持多 app，可设置为数组 array('app','app2','app3');默认第一个 app,可在 url 中用&app=app2 来读取其它 app，如果 rewrite 请选用模式 5 自定义 rewrite
3、mysqli bind 复杂 sql 语句的支持

如有疑问联系作者!

附件：

从 v1.0 版本升到 v2.0 版本说明

更改 index.php
v1.0 的
＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
// 定义 科亿 php 框架路径
define('KY_PATH', '../frm');
//定义项目名称和路径
define('APP_PATH', '.');
// 加载框架入口文件
require(KY_PATH."/kyphp.php");
//实例化一个网站应用实例
＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

v2.0
==========
// 定义 科亿 php 框架路径
define('APP_PATH',dirname(__FILE__));//当前项目路径
define('KYPHP_PATH',APP_PATH.'/../../kyphp/');//KYPHP 框架路径
define('__CHARSET__','gbk');
require KYPHP_PATH."kyphp.php";
//如果要自定义 config 路径，需要定义__CONFIG__
//define('__CONFIG__',APP_PATH.'/config.php');
//$config=require(__CONFIG__);
KYPHP::Run(); //自定义 config 请定义后用 KYPHP::Run($config); index 和 config 可合并为同一文件
==========

config.php 中增加
==========
'DEFAULT_C_PATH' =>'lib', //已更改为 controller
'app_dir' =>'.', //已更改为 app

==========
2.0 不再支持$mysql 全局及 fetch_array