源码框架开发使用文档	
框架原型	Flight-php
框架版本	3.0
框架开发	孙一
文档撰写	孙一
应用范围	新浪支付中心【PHP 开发组】
版本改动	1：升级数据模型基类 2：通过 HTTP 接口统一记录和管理日志 3：开发 DEMO 模块作为示例
日　　期	2015/8/20

目录

一：框架简介

基于新浪支付中心的业务需求开发本框架，支持独立 CGI 接口与 MVC 框架开发，框架的原型是 Flight-PHP。由于项目将部署的环境无法支持本地写操作，故用接口方式记录日志。

二：框架安装

配 HOST 域名：124.238.233.103 sunyi.sinapay.dev.com
框架下载 URL：http://sunyi.sinapay.dev.com/sinaWork3.0
框架安装方法：将框架压缩包解压至开发目录
框架默认使用 Apache-WEB 服务器
如果安装后无法使用
请修改 Apache 配置文件 httpd.conf,使支持 Rewrite 重定向
对于 nginx-WEB 服务器请参考 Flight-PHP 安装手册 http://flightphp.com/
框架结构如图 3-1：

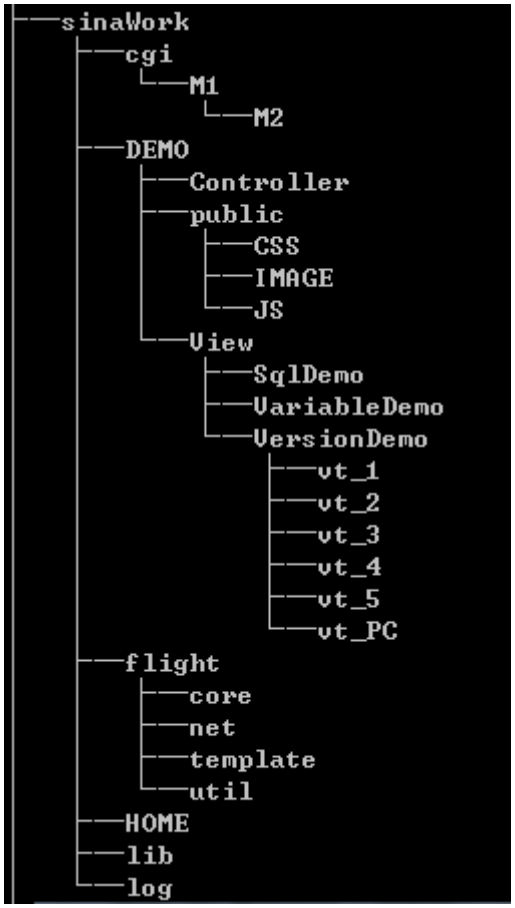


图 3-1

目录结构说明（详细使用方法见下文）：

1. cgi ： 在此文件夹下开发 HTTP 接口类。
M1/M2…接口所属的各级业务，最多允许三层

2. DEMO ： 框架使用的示例模块
3. HOME ： MODULE 缺省时默认模块
4. flight ： flight-php 框架文件（勿修改）。
5. Controller ： 在此文件夹下开发控制器。
6. View ： 在此文件夹下开发视图。
7. Public ： 加载 CSS、JS 等代码，添加图片、音乐等资源，加载用户自定义类与函数。
8. Lib ： flight-php 框架扩展文件（勿轻易修改）。

三： HTTP-CGI 接口开发

3.1：路由

路由规则：/sinaWork/cgi/[@M1]/[@M2]/[@M3]/@space-@action
其中=>
[@M1]/[@M2]/[@M3]分别表示第一、二、三级业务（可选）
@space:cgi 所属模块
@action:cgi 操作接口
路由示例：
http://@webroot/sinaWork/cgi/M1/M2/CgiTest-cgiAction
于是将访问 CgiTest 模块中的 cgiAction 操作接口。

3.2：代码示例

在框架根目录 cgi 文件夹下建立/M1/M2/CgiTest.class.php：
写入示例代码 3-1。

```php
<?php
class CgiTest extends CgiBase
{
    function cgi_1()
    {
        $this->answer = array( 'status' => 0 , 'answer' => 1 );
    }
    function cgi_2()
    {
        $this->answer = array( 'status' => 0 , 'answer' => 2 );
    }
    function cgiAction()
    {
        $this->answer = array( 'status' => 0 , 'answer' => 'Hello! cgiAction working!' );
    }
`
```

代码 3-1

使用 http://@webroot/sinaWork/cgi/M1/M2/CgiTest-cgiAction 访问见图 3-2：

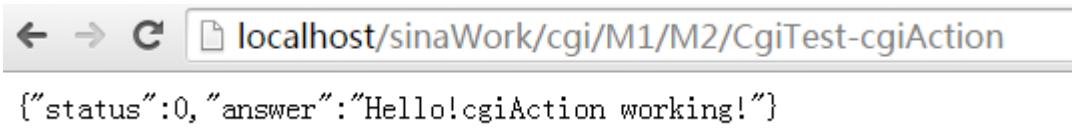


图 3-2

如果 cgi 文件夹下没有 CgiTest.class.php 或 CgiTest 类中没有 cgiAction 函数，均会返回错误。

3.3：接口返回规则

接口支持三种返回格式：

1：JSON 返回格式

2：XML 返回格式

3：文本(或自定义)返回格式

如果未指定返回格式，则框架默认返回 JSON 格式数据。

指定返回格式方法：在 CGI 的 URL 添加参数：?returnType=JSON/XML/TEXT

对于 JSON/XML 接口：

开发人员把输出结果存入**$this->answer** 数组，使用方法见代码 3-1。

框架为保证输出合法 JSON/XML 数据，将清空用户在测试过程中输出的调试信息，开发人员可以使用文本输出模式进行调试。

如图 3-3 为设定返回 XML 格式时返回展示

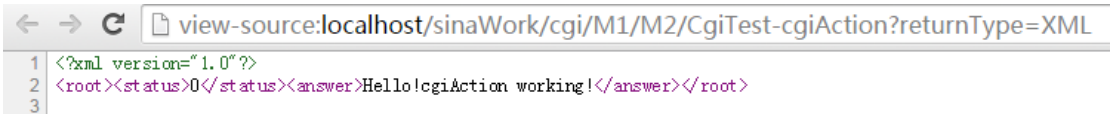


图 3-3

3.4：http 响应码

执行：$this->headerHttp($number)设定 http 响应码

响应码对应：

100 => "HTTP/1.1 100 Continue",
101 => "HTTP/1.1 101 Switching Protocols",
200 => "HTTP/1.1 200 OK",
201 => "HTTP/1.1 201 Created",
202 => "HTTP/1.1 202 Accepted",
203 => "HTTP/1.1 203 Non-Authoritative Information",
204 => "HTTP/1.1 204 No Content",
205 => "HTTP/1.1 205 Reset Content",
206 => "HTTP/1.1 206 Partial Content",
300 => "HTTP/1.1 300 Multiple Choices",
301 => "HTTP/1.1 301 Moved Permanently",
302 => "HTTP/1.1 302 Found",

303 => "HTTP/1.1 303 See Other",
304 => "HTTP/1.1 304 Not Modified",
305 => "HTTP/1.1 305 Use Proxy",
307 => "HTTP/1.1 307 Temporary Redirect",
400 => "HTTP/1.1 400 Bad Request",
401 => "HTTP/1.1 401 Unauthorized",
402 => "HTTP/1.1 402 Payment Required",
403 => "HTTP/1.1 403 Forbidden",
404 => "HTTP/1.1 404 Not Found",
405 => "HTTP/1.1 405 Method Not Allowed",
406 => "HTTP/1.1 406 Not Acceptable",
407 => "HTTP/1.1 407 Proxy Authentication Required",
408 => "HTTP/1.1 408 Request Time-out",
409 => "HTTP/1.1 409 Conflict",
410 => "HTTP/1.1 410 Gone",
411 => "HTTP/1.1 411 Length Required",
412 => "HTTP/1.1 412 Precondition Failed",
413 => "HTTP/1.1 413 Request Entity Too Large",
414 => "HTTP/1.1 414 Request-URI Too Large",
415 => "HTTP/1.1 415 Unsupported Media Type",
416 => "HTTP/1.1 416 Requested range not satisfiable",
417 => "HTTP/1.1 417 Expectation Failed",
500 => "HTTP/1.1 500 Internal Server Error",
501 => "HTTP/1.1 501 Not Implemented",
502 => "HTTP/1.1 502 Bad Gateway",
503 => "HTTP/1.1 503 Service Unavailable",
504 => "HTTP/1.1 504 Gateway Time-out"

3.5：记录日志

开发人员记录日志，在任意地方调用
WriteLog (“logmsg”)
框架将连接日志服务器
将日志记录在对应的日志目录下作为开发者自定义的日志【DEV_LOGS】
日志分类“

1：业务自动记录的日志【AUTO_LOGS】
    1.1：CGI 接口访问入口和结果日志
    1.2：数据库 SQL 访问记录的日志
2：开发者自定义的日志【DEV_LOGS】
    2.1 开发者在 CGI 接口逻辑中输出的自定义日志
    2.2 开发者在 MVC 业务逻辑中输出的自定义日志

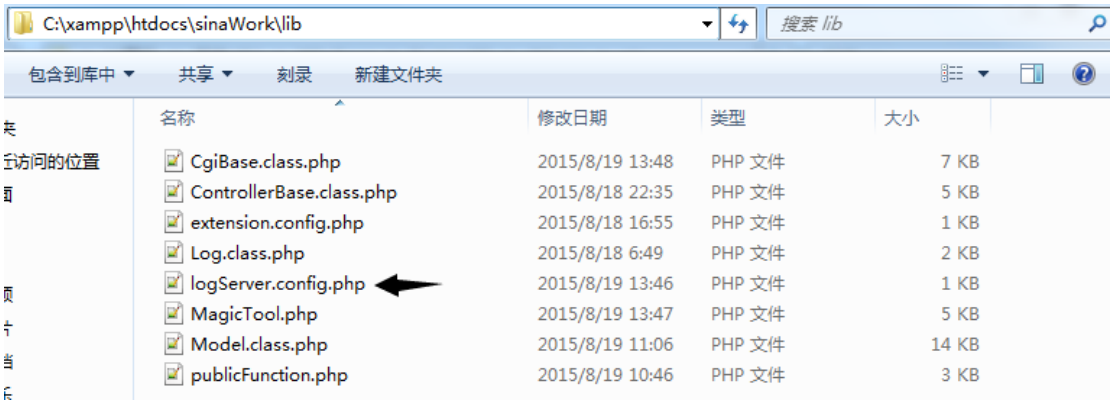
日志系统目录结构：

```
LOGS
    AUTO_LOGS
        CGI_LOGS
            2015-08-19
                CgiTest-cgiAction.log
        MYSQL_LOGS
            2015-08-19
                MVC-DEMO-SqlDemo-insert.log
    DEV_LOGS
        2015-08-19
            CGI-CgiTest-cgiAction.log
```

可以通过修改框架内的配置文件设置日志服务器信息。

打开 sinaWork/Lib 的 logServer.config.php

修改 URL 和 IP 为日志包部署的路径




```php
<?php
return array(
        'url'=>'http://[redacted].sinapay.dev.com/LogRabbit/logDog.php',
        'ip' =>'124.238.233.103');
```

四：数据模型（数据库查询）

当前版本的框架封装了 Mysql 数据库的连接与查询。
支持单表插入、查询、修改的操作。
支持 order by、group by、limit 等查询限制。
在使用数据库操作时，代码任何位置实例化 Model 对象
示例：
连接数据库：

```
$myModel = new Model("127.0.0.1",3306,"username","password","user");
```

查询：

```
$myModel->select();
```

$myModel->count();
修改：
$myModel->update();
插入：
$myModel->insert();
设定条件/键值对：（下面省略$myModel->）
setField($field) => 设定待查询的字段， 传入参数为数组；
setWhere($field,$op,$value) => 添加查询/修改条件， 传入（字段，比较符，值）；
允许多次调用添加多个条件，使用见下面例子

```
/**************************************************
    * 设定 where 条件
    * 方法一:注意累计添加
    *    field ：字段
    *    op      ：操作符（=,>,<,=,<=,>=,in.....）
    *    value   : 值
    *    @EXAMPLE
    *    $this->setWhere('age','<',20)
    * 方法二：以数组形式统一添加
    *    @EXAMPLE
    *    $this->setWhere(array(array('age','<',27),
                                array('age','>',20)));
    * 方法三：直接存入完整 WHERE 语句
    */
```

setPage($page) => 设定查询页码 //默认查询第 0 页（即第 1 页）；
setLimit($limit) => 设定每页数据条数 //默认为 20，设定为 0 不限制；
setOrder($field,$isDESC) => 添加 order by 逻辑， 传入（字段，是否 DESC）；
setGroup($field) => 添加 group by 多级；
setKeyValue($key,$value) => 添加键值对：通用于 update 和 insert 查询
setKeyValues($kv) => 添加键值对数组：通用于 update 和 insert 查询
数组结构：
Array(k1 => v1 , k2 => v2 ……)

在执行一次增查改逻辑之后，如果要执行其它 sql 逻辑，不建议重新实例化新的 model 对象。
可以使用：$myModel->clear(); 清空对象配置
Clear 可以带有参数，指定具体清除哪一项配置
【当前版本关闭此功能，每次执行语句结束，自动清除属性设定】
可选参数[直接贴函数代码了^_^]：

```php
//////////////////////////////////////////////////////////////////////////
public function clear($opr = 'all')
{
    switch ($opr)
    {
        case 'table'    : $this->table    = null;       break;
        case 'field'    : $this->field    = array();    break;
        case 'where'    : $this->where    = array();    break;
        case 'page'     : $this->page     = 0;          break;
        case 'limit'    : $this->limit    = 20;         break;
        case 'order'    : $this->order    = array();    break;
        case 'group'    : $this->group    = array();    break;
        case 'keyValue' : $this->keyValue = array();    break;
        case 'all':
        default:
            $this->table      = null;
            $this->field      = array();
            $this->where      = array();
            $this->page       = 0;
            $this->limit      = 20;
            $this->order      = array();
            $this->group      = array();
            $this->keyValue   = array();
    }
}
//////////////////////////////////////////////////////////////////////////
```

SQL 脚本硬性查询调用：

因为框架模型类并不支持删除、多表查询、嵌套查询、批量查询等操作，故提供通用的 SQL 硬性查询函数：

$myModel->HARDQUERY($query)

其中$query 为待执行的 SQL 查询语句。

五：HTTP 接口访问方法

```
/**************************************************

    * HTTP 接口调用请求方法

    * url       :带请求接口的 url

    * opt           :请求参数,数组(可选)

    *              {

    *                  timeout          :超时时间

    *                  method           :GET/POST

    *                  data             :类型为 string 或 array，
```

```
     *                  ip                  :域名对应的 ip，指定 ip 后，url 中的域名将不依赖 host 和域名解析
     *                  decode              : true/false ，是否对结果进行 json_decode
     *                  check_callback      :检查结果回调函数,例:function check($ret_data);
     *              }
     * @example
     * $url = "http://c.isd.com/AppCmdb/interface/cmdbInterface1.php"
     * $opt = array(
     *              'ip'        => '10.137.155.40',
     *              'data'          => json_encode(array('xxx'=>'xx')),
     *              "method"    => "POST",
     *              "timeout"=> 60,
     *              "decode"    => true,
     *            //"headers"=> ""
     * );
     * $ret = $this->http_request($url,$opt);
     */
```

六：基于 MVC 架构的网站开发

6.1：路由

http://@webroot/sinaWork/[@module/]@controller-@action
其中=>
@module:　　模块[可选项]，默认 HOME
@controller:　　控制器类名
@action:　　控制器类内动作函数名
将访问@module 模块下，@controller 控制器的@action 动作函数

6.2：代码示例

开发人员在 @webroot\sinaWork\@module\Controller 下建立控制器类。
@webroot\sinaWork\@module\view\@controllor 下建立视图文件

控制器类名与类文件名保持一直

控制器类文件名： @controller.class.php

视图文件名为： @action.php

见代码 6-1 ; 6-2：
//webroot\sinaWork\HOME\Controller\C_test.class.php

```php
<?php
class C_test extends    ControllerBase
{
    function __construct()
    {
        parent::__construct();
    }
    public function A_test()
    {
        echo "come into A_test<br>";
        $this->setVar("varToView",date('Y-m-j h:i:s A'));
    }
}
```

代码 6-1

//webroot\sinaWork\HOME\View\C_test\A_test.php

```php
<?php
echo $varToView;
```

代码 6-2

访问 url: http://@webroot/sinaWork/C_test-A_test
运行结果：

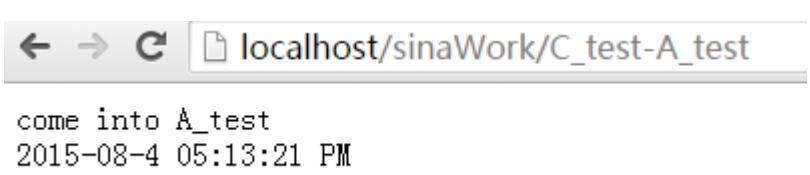


图 6-1

6.3：数据传递至 VIEW

在控制器内处理业务逻辑之后，得到的业务数据可以通过两种方法传递给 View

1：传递 PHP 变量 ：$this->setVar(变量名,数据);　　//支持数组

2：传递 JS 变量　：$this->setJson(变量名,数据);　　//支持数组

七：常量使用

全局常量：

```
$_ROOT_;          //框架根目录
$_MODULE;         //所属模块
$_CONTROLLER;     //控制器名
$_ACTION;         //动作名
$_PUBLIC_;        //默认 public 文件路径
$_CSS_;           //默认 CSS 文件路径
$_JS_;            //默认 JS 文件路径
$_IMAGE_;         //默认图片文件路径
$_ROOT_;          //框架根目录
```

Controller 类内部常量：

```
    $this->GET      //存储$_GET
    $this->POST     //存储$_POST
```