

用 PHP 开始你的 MVC (一)整合你的站点入口

文章整理: [朱英](#) 文章来源: 网络

这是一篇介绍如何用 php 来实现 MVC 模式开发的文件。关于 MVC 模式的技
术文章网上随处可以, 所以这篇文件将不再讲述这种模式的优缺点(实际上
是我说不清楚), 只讲他的 php 技术实现。并且在以后的系列文章中也是以讲技术为主。

一、实现统一的网站入口 (在 MVC 中调用 Controller 层的方法, 也就是控制层)

大家也许经常在网上看到这样的路径(<http://www.aaa.com/aaa/bbb/aaa?id=5>), 让人不解, 这样的网站的实现方式有几种可能性:

- 1、隐藏文件的扩展名, 对这种做法的好处, 众说纷纭, 不过个人觉得没有必要;
- 2、用了网站的重定向规则, 实现虚拟路径;
- 3、强制文件解析的方式, 实现虚拟路径。

用第 2\3 种方法可以实现网站的统一接口, 合理的整合网站, 更好的体现网站的安全性和架构, 用这两种方式的网站大多是使用“MVC”模式构

建和实现的。

下面是一个例子

访问路径如下:

```
...../test/********/Bad  
...../test/********/Good  
(其中的“*****”可以用任何字符串替换, “.....”是你的 web 路径)
```

文件的目录结构如下

```
|-- .htaccess  
|-- test  
|-- Application.php  
|-- Controller/GoodController.php  
|-- Controller/BadController.php
```

注意 文件“.htaccess”, 在 windows 下不能直接建立的, 可以在命令行模式下建立。

文件 0:(.htaccess) (这个文件是更改 apache 的配置方式用的)

```
<files test>  
force-type application/x-httpd-php  
</files>
```

文件 1: (test.php)

```
<?php
/*
 * test.php
 *
 * 作为你的网站的入口的文件
 * 用来初始化和入口
 * 调用执行 Controller 的调用
 *
 */
require "Application.php";
$aa = new Application();
$aa->parse();
$aa->go();

?>
```

文件 2: (GoodController.php)

```
<?php
/*
 * GoodController.php
 *
 * 用来控制 url=/test/Good 来的访问
 *
 */
class GoodController{
    /*
     * 控制类的调用方法，唯一的报漏给外部的接口
     */
    function control() {
        echo "this is from GoodController url=*****/test/Good";
    }
}

?>
```

文件 3: (BadController.php)

```
<?php
/*
 * BadController.php
 */
```

```
*  
* 用来控制 url=/test/Bad 来的访问  
*  
-----*/  
class BadController{  
    /*  
     * 控制类的调用方法，唯一的报漏给外部的接口  
     */  
    function control(){  
        echo "this is from GoodController url=*****/test/Bad";  
    }  
}  
  
?>
```

文件 4: (Application.php)

```
<?php  
/*-----  
 * Application.php  
 *  
 * 用来实现网站的统一入口，调用 Controller 类  
 *-----*/  
class Application{  
    //用来记录所要进行的操作  
    var $action;  
    //controller 文件的路径名  
    var $controllerFile;  
    //controller 的类名  
    var $controllerClass;  
  
    function Application(){  
    }  
  
    function parse(){  
        $this->_parsePath();  
        $this->_getControllerFile();  
        $this->_getControllerClassname();  
    }  
    /*  
     * 解析当前的访问路径，得到要进行动作  
     */  
    function _parsePath(){
```

```
list($path, $param) = explode("?", $_SERVER["REQUEST_URI"]);
    $pos = strrpos($path, "/");
    $this->action = substr($path, $pos+1);
}
/*
 * 通过动作$action, 解析得到该$action 要用到的 controller 文件的路径
 */
function _getControllerFile() {
    $this->controllerFile = "./Controller/".$this->action."Controller.php";
    if(!file_exists($this->controllerFile))
        die("Controller 文件名(\".$this->controllerFile.\")解析错误");
    require_once $this->controllerFile;
}
/*
 * 通过动作$action, 解析得到该$action 要用到的 controller 类名
 */
function _getControllerClassname() {
    $this->controllerClass = $this->action."Controller";
    if(!class_exists($this->controllerClass))
        die("Controller 类名(\".$this->controllerClass.\")解析错误");
}
/*
 * 调用 controller, 执行 controller 的动作
 */
function go() {
    $c = new $this->controllerClass();
    $c->control();
}
?
?>
```

用 PHP 开始你的 MVC (二) 抽象数据库接口

文章整理: [朱英](#) 文章来源: 网络

二、抽象数据库接口(利用数据操作管理类)

在用 mvc 模式开发的时候, model 层负责数据库的所有操作, 为了对数据库的操作进行统一的管理, 我们需要定义一个数据库操作管理类, 由他来接替所有的数据库操作, 也就是整个系统中只有这个数据库操作管理类, 可以直接对数据库进行操作, 如果其他的类需要对数据库进行操作, 那它就必须调用和通过这个类来实现。

下面的 Db 类就是一个这样的类。

```
<?php
/*
 *      descript : mysql 数据库操作管理类
 */
/*
 * example 1:  获取序列
 *   <?
 *   $db = new Db();
 *   $result = $db->getSeq('' art_id'', 2, 1);
 *   ?>
 */
/*
 * example 2:  分页查询
 *   <?
 *   $db = new Db();
 *   $result = $db->&queryPage('' select id, name from table'', 2, 10,
DB_FETCH_ASSOC);
 *   foreach($result as $row)
 *       echo $row['' id''], '' -----''. $row['' name'']. '' <br>'';
 *   ?>
 */
/*
 * example 3:  插入数据
 *   <?
 *   $db = new Db();
 *   $result = $db->execute("insert into table (id, name)
values(1, '' name'')");
 *   if($result>0)      echo '' 插入数据成功'';
 *   else                  echo '' 插入数据失败'';
 *   ?>
 */
/*
 * 定义数据库联接选项
 * @var DB_HOST      string    数据库主机名称或地址
 * @var DB_NAME      string    数据库名称
 * @var DB_USER      string    数据库用户名
 * @var DB_PWD       string    数据库用户的密码
 * @var DB_PCONNECT  boolean   是否建立持久连接
 */
define('' DB_HOST'', '' localhost'');
define('' DB_NAME'', '' test'');
define('' DB_USER'', '' root'');
```

```
define( 'DB_PWD' , '''' );
define( 'DB_PCONNECT' , true );
/*
 * 定义返回数据查询结果的类型
 * @var DB_FETCH_ASSOC int 结果调用方式: $result['name']
 * @var DB_FETCH_NUM int 结果调用方式: $result[0]
 * @var DB_FETCH_BOTH int 结果调用方式: $result['name'] 或
$result[0]
 * @var DB_FETCH_OBJECT int 结果调用方式: $result->name
*/
define( 'DB_FETCH_ASSOC' , 0 );
define( 'DB_FETCH_NUM' , 1 );
define( 'DB_FETCH_BOTH' , 2 );
define( 'DB_FETCH_OBJECT' , 3 );
/*
 * 定义默认序列发生器的名称
 */
define( 'DB_SEQUENCE_TABLENAME' , 'sequences' );

class Db{
/*
 * 当前数据库联接选项
 */
var $dbHost = DB_HOST;
var $dbName = DB_NAME;
var $dbType = 'Mysql';
var $dbUser = DB_USER;
var $dbPwd = DB_PWD;
var $pcnn = DB_PCONNECT;
/*
 * 当前数据库连接
 */
var $cnn = '';
/*
 * 数据查询结果的返回类型
 */
var $queryFetchType = DB_FETCH_ASSOC;
/*
 * 初始化函数
 */
function Db() {
    $this->cnn = ($this->pcnn? mysql_connect($this->dbHost,
$this->dbUser, $this->dbPwd):
mysql_connec
```

```
t($this->dbHost, $this->dbUser, $this->dbPwd)) or
                                         $this->_halt(
,, 数据库连接错误');
    mysql_select_db($this->dbName, $this->cnn) or $this->_halt('
数据库选择错误');
}

/*
 *  数据查询函数
 *
 * @param $sql      string  数据查询语句
 * @param $fetchType int    数据查询结果的返回类型
 *
 * @return          array   数据查询结果
 */
function &query($sql, $fetchType=DB_FETCH_ASSOC) {
    $data = array();
    $rs = &mysql_query($sql, $this->cnn) or $this->_halt(' 数据查
询错误', $sql);
    $exe = $this->_getCommand($fetchType);
    while($row=&$exe($rs))
        $data[] = &$row;
    return $data;
}
/*
 *  分页数据查询函数
 *
 * @param $sql      string  数据查询语句
 * @param $page     int    当前预查询页码
 * @param $pageSize int    每页显示多少条纪录
 * @param $fetchType int    数据查询结果的返回类型
 *
 *      数据查询结果, 以及数据的分页信息
 * @return      array('pageSize' => 每页显示的条数
 *                   , 'recordCount' => 总纪录数
 *                   , 'pageCount' => 总页数
 *                   , 'page' => 当前页码
 *                   , 'isFirst' => 是否第一页
 *                   , 'isLast' => 是否最后一页
 *                   , 'start' => 返回结果的第一条纪录的序号
 *                   , 'sql' => 查询的 sql 语句
 *                   , 'data' => 查询得到的数据结果
 * )
 *      数据查询结果, 以及数据的分页信息
*/

```

```

        function &queryPage($sql, $page=1, $pageSize=20,
$fetchType=DB_FETCH_ASSOC) {
            $countSql = preg_replace('' | SELECT.*FROM|i'', '' SELECT COUNT(*)
count FROM'', $sql);
            $data['' pageSize''] = (int)$pageSize<1? 20: (int)$pageSize;
            $data['' recordCount''] = $this->getOne($countSql, '' count'');
            $data['' pageCount''] =
ceil($data['' recordCount'']/ $data['' pageSize'']);
            $data['' page''] = $data['' pageCount''] ==0? 0: ((int)$page<1? 1:
(int)$page);
            $data['' page''] = $data['' page'']>$data['' pageCount'']?
$data['' pageCount'']:$data['' page''];
            $data['' isFirst''] = $data['' page'']>1? false: true;
            $data['' isLast''] = $data['' page'']<$data['' pageCount'']?
false: true;
            $data['' start''] = ($data['' page''] ==0)? 1:
($data['' page'']-1)*$data['' pageSize'']+1;
            $data['' sql''] = $sql.'' LIMIT
''.($data['' start'']-1).'', ''.$data['' pageSize''];
            $data['' data''] = &$this->query($data['' sql''], $fetchType);
            return $data;
        }
/*
 *  进行数据查询只返回第 1 行的数据
 *
 * @param $sql          string    数据查询语句
 * @param $fetchType    int      数据查询结果的返回类型
 *
 * @return             array    数据查询结果
 */
function &queryRow($sql, $fetchType=DB_FETCH_ASSOC) {
    $rs = &mysql_query($sql, $this->cnn) or $this->_halt('' 单行数
据查询错误'', $sql);
    $exe = $this->_getCommand($fetchType);
    return $exe($rs);
}
/*
 *  进行数据查询只返回第 1 行第 n 列的数据
 *
 * @param $sql          string    数据查询语句
 * @param $field         int      返回数据列的名称 或 数字序号
 *
 * @return             string   返回单个字段的值
*/

```

```
function &getOne($sql, $field = 0) {
    $rs = &mysql_query($sql, $this->cnn) or $this->_halt(''单个数据查询错误'', $sql);
    $row = mysql_fetch_array($rs);
    return $row[$field];
}
/*
 * 进行 sql 语句, 包含 DELECT / INSERT / UPDATE..... 的执行语句
 *
 * @param $sql          string  数据查询语句
 *
 * @return             string  返回该语句影响的数据行数
 */
function execute($sql) {
    $rs = mysql_query($sql) or $this->_halt(''语句执行错误'', $sql);
    return mysql_affected_rows($this->cnn);
}
/*
 * 得到最后一次插入数据的编号
 */
function getInsertId() {
    return mysql_insert_id($this->cnn);
}
/*
 * 序列发生器, 用来生成不重复的序列值
 *
 * @param $fieldName      string  序列的名称
 * @param $step            int     序列号间隔
 * @param $start           int     序列号的起始数值
 *
 * @return                int     新的序列值
 */
function getSeq($fieldName, $step=1, $start=1) {
    $table = DB_SEQUENCE_TABLENAME;
    $step = (int)$step;
    $start = (int)$start;
    $rs = mysql_query("UPDATE $table SET seq_num=seq_num+($step)
WHERE seq_name=''$fieldName''");
    if (!$rs || mysql_affected_rows($this->cnn)<1) {
        $rs = mysql_query('' SELECT * FROM '' . DB_SEQUENCE_TABLENAME,
$this->cnn);
        if (!$rs) {
            $sql = "CREATE TABLE $table (
```

```
        seq_name VARCHAR( 20 ) NOT NULL ,
        seq_num BIGINT( 20 ) DEFAULT 1 NOT NULL ,
        PRIMARY KEY (seq_name))";
$rs = mysql_query($sql) or $this->_halt('创建序列发生器表失败', $sql);
}
$rs = mysql_query("INSERT INTO $table
VALUES(''$fieldName'', $start)") or
        $this->_halt('添加新序列错误', $sql);
$seq = $start;
} else {
        $seq = &$this->getOne("SELECT seq_num FROM $table WHERE
seq_name=''$fieldName''");
}
return $seq;
}

function _getCommand($fetchType) {
switch($fetchType) {
    case DB_FETCH_ASSOC: $exe = ''mysql_fetch_assoc''; break;
    case DB_FETCH_NUM: $exe = ''mysql_fetch_row''; break;
    case DB_FETCH_BOTH: $exe = ''mysql_fetch_array''; break;
    case DB_FETCH_OBJECT: $exe = ''mysql_fetch_object''; break;
    default: $exe = ''mysql_fetch_array''; break;
}
return $exe;
}

function _halt($msg) {
$errNo = mysql_errno($this->cnn);
$errStr = mysql_error($this->cnn);
die("数据库错误: $msg<br> $errNo : $errStr");
}
}
?>
```

[点击这里进入对应文章地址](#) [\[进入主站\]](#)

用 PHP 开始你的 MVC(三) 实现你的 Model 层

文章来源：[朱英](#) 文章来源：网络

三、实现你的 Model 层

Model 层，就是 MVC 模式中的数据处理层，用来进行数据和商业逻辑的装封，进行他的设计的时候设计到三个概念：

-----Model 类。是实体类。用来保存数据库表格中一条记录的所有字段的数据。并且可以验证这条记录数据的完整性。

-----ModelManager 类。是实体类的管理类。通常每一个实体类(Model)都要有一个对应的管理类(ModelManager)。管理类可以用来管理实体类里面的数据纪录(例如删除/添加/更改.....)。但是 ModelManager 类不一定要有对应的 Model 类。

-----db 类。用来管理对数据库的联接。ModelManager 类所有的对数据的操作。都是通过这个 db 类来实现的。在整个 MVC 模式中。只有这个 db 类可以直接对数据库进行操作。同时也只有 ModelManager 类可以对 db 类进行调用。

看上去好象是比较麻烦。但是实际上并不复杂。这种 Model 层设计方式。和网上购物系统的购物车程序是极其相似的。Model 可以看作是购物车里的单个商品的信息类。Manager 可以看作是订单。订单是用来管理采购的商品的。

下面是一个简单的例子。应该是比较典型的。着重看他的整个设计和流程的实现。仔细研究一下。其实不难。

注意：下面例子使用的所有的类和方法都是经过简化的。实际情况比这个要复杂的多。但是。作为一个实例已经是足够用了。

文件夹结构：

```
| - Db.php
| - Model.php
| - Manager.php
| - ModelTest1.php
| - ModelTest2.php
| - ModelTest3.php
| - ModelTest4.php
| - Model /
|   | - Model / ClassModel.php
|   | - Model / StudentModel.php
|   | - Model / ClassManager.php
|   | - Model / StudentManager.php
```

注意文件夹和文件名的大小写

内容：假设有一个数据库，保存在两张表，一张是 class(班级) 表格，一张是 student(学生) 的表格，

```
class 表格字段: cls_id-----int-----not null
                  cls_name-----string-----not null
                  cls_address----string----null

student 表格字段:stu_id-----int-----not null
                  stu_clsid-----int-----not null
                  stu_name-----string-----null
```

ClassModel.php 里面是 class 表的一个实体类 ClassModel
 ClassManager.php 里面是 ClassModel 的管理类 ClassManager
 StudentModel.php 里面是 student 表的一个实体类 StudentModel
 StudentManager.php 里面是 StudentModel 的管理类 StudentManager
 Db.php 里面是一个数据库操作管理类, 他和里面用的接口和正常使用情况是一样的, 但是本例只是模拟的实现了这个借口. 因此, 可以在不用真实数据库的情况下运行.

文件 0: (Model.php) Model 层实体的基础类

```
<?php
//用来包装信息实体的基础类
class Model{
    //这个实体类的数据,
    //example: array("id"=>1, "name"=>"this is name");
    var $data;
    //这个实体类的数据约束信息, 用来判断加入的$data 数据的准确性
    //see: ClassModel
    var $match;
    //与该实体对应的数据库中表的名称
    var $table;
    //初始化
    function Model(&$data) {
        $this->data = &$data;
    }
    //设置该实体的某个数据是值
    function set($key, $value) {
        $this->data[$key] = $value;
    }
    //获取该实体的某个数据
    function get($key) {
        return $this->data[$key];
    }
}
```

```

}

//获取该实体的全部数据
function getData() {
    return $this->data;
}

//获取该实体的约束信息
function getMatch() {
    return $this->match;
}

//验证实体数据的准确性和完整性
function isValid() {
    foreach($this->match as $key=>$value) {
        if(!isset($value["null"]) && !isset($this->data[$key])) die("$key 的数值不能为空");
        //.....可以在加其他的判断，例如是否超过如许的最大数值，或
        长度过长.....
    }
}
?>

```

文件 1: (Manager.php)Model 层进行实体管理的基础类

```

<?php
//对实体信息进行管理的基础类
class Manager{
    //数据库管理类对象
    var $db;
    //初始化
    function Manager() {
        $this->db = new Db();
    }
    //用来向数据库中插入实体信息
    function insert(&$model) {
        $model->isValid();
        $table = $model->table;
        $match = $model->getMatch();
        $data = $model->getData();
        $str1 = $str2 = array();
        foreach($match as $key=>$value) {
            if(isset($data[$key])) {
                $str1[] = $key;
                $str2[] = ($value["type"]=="C")? "\"". $data[$key]. "\""
            }
        }
        $sql = "insert into $table($str1) values($str2)";
        $this->db->query($sql);
    }
}

```

```
" : $data[$key];
        }
    }
    $sql = "INSERT INTO $table (" . implode(", ", $str1) . ") VALUES(" .
        implode(", ", $str2) . ")";
    return $this->db->execute($sql);
}
?>
```

文件 2: (ClassModel.php) 班级信息的实体类

```
<?php
//用来包装班级信息的实体类
class ClassModel extends Model {

    var $data = array();
    //在 $match 中,
    //type 用来表示数据的类型 (I 表示整数, C 表示字符串)
    //name 用来表示在数据库表中的字段名
    //null 表示该字段的值是否准许为空
    //      (数组中有"null"=>true 表示是准许为空, 否则不能为空)
    var $match = array("cls_id" => array("name"=>"cls_id", "type"=>"I"),
                       "cls_name" => array("name"=>"cls_name", "type"=>"C"),
                       "cls_address" => array("name"=>"cls_address",
                                         "type"=>"C", "null"=>true)
                     );

    var $table = "class";
    //初始化
    function ClassModel(&$data) {
        parent::Model($data);
    }
    //用来获取这个班级的学生的信息
    function getStudent() {
        require_once "./Model/StudentManager.php";
        $manager = new StudentManager();
        $classId = $this->get("cls_id");
        return $manager->getList($classId);
    }
}
```

?>

文件 3: (StudentModel.php) 学生信息的实体类

```
<?php
//用来包装学生信息的实体类
class StudentModel extends Model {

    var $data = array();
    //在 $match 中,
    //type 用来表示数据的类型 (I 表示整数, C 表示字符串)
    //name 用来表示在数据库表中的字段名
    //null 表示该字段的值是否准许为空
    //      (数组中有"null"=>true 表示是准许为空, 否则不能为空)
    var $match = array("stu_id" => array("name"=>"stu_id", "type"=>"I"),
        "stu_clsid" => array("name"=>"stu_clsid", "type"=>"I"),
        "stu_name" => array("name"=>"stu_name", "type"=>"C", "null"=>true)
    );

    var $table = "student";
    //初始化
    function StudentModel(&$data) {
        parent::Model($data);
    }
}
?>
```

文件 4: (ClassManager.php) 班级实体的管理类

```
<?php
//班级实体信息的管理类
class ClassModelManager extends Manager {
    //初始化
    function ClassModelManager() {
        parent::Manager();
    }
    //获取班级列表
    function &getList() {
        $sql = "SELECT * FROM class";
```

```

        return $this->db->query($sql);
    }
    //查找并返回一个班级的实体类
    function &findOneModel($id) {
        $sql = "SELECT * FROM class WHERE cls_id=$id";
        $data = $this->db->getOne($sql);
        if($data==null) die("该班级不存在！");
        require_once "./Model/ClassModel.php";
        $model = new ClassModel($data);
        return $model;
    }
}
?>
```

文件 5: (StudentManager.php) 学生实体的管理类

```

<?php
//学生信息实体的管理类
class StudentManager extends Manager{
    //初始化
function StudentManager() {
    parent::Manager();
}
//获取某个班级的学生的列表
function &getList($classId) {
    $sql = "SELECT * FROM student WHERE stu_clsid=$classId";
    return $this->db->query($sql);
}
}
?>
```

文件 6: (Db.php) 数据库联接管理类，用于共享并管理数据的访问。由于这个类涉及的内容不是本章要讨论的内容，所以这个类模拟了“真实的数据库管理类的方法”，借口是和正常的类是一样的，但是接口函数里面的内容是不对的，只是模拟的数据。网上有很多这种类的做法，可以自己到晚上找找，（**另外本系列文章的第二章里也有详细的介绍**）。

```

<?php
//数据库操作管理类
class Db{
    //数据库联接
    var $con;
```

```

//初始化
function Db() {
    // $this->con=mysql_connect (*****); .....
}

//执行数据查询语句
function &query($sql) {
    //$result = mysql_query($sql); .....
    //return $result;
    if($sql=="SELECT * FROM student WHERE stu_clsid=2")
        return array("0"=>array("stu_id"=>1, "stu_clsid"=>2, "stu_name"=>"student1"),
                     "1"=>array("stu_id"=>2, "stu_clsid"=>2, "stu_name"=>"student2"))
    );
    die("空班级");
}

//获取一条数查询结果
function getOne($sql) {
    //$result = mysql_query($sql); .....
    //return $result[0];
    if($sql=="SELECT * FORM class WHERE cls_id=1")
        return null;
    if($sql=="SELECT * FORM class WHERE cls_id=2")
        return array("cls_id"=>2, "cls_name"=>"classname", "cls_address"=>"classaddress");
    }

//执行数据库更新/添加/删除语句
function execute($sql) {
    //mysql_query($sql);
    echo "<br>正在进行插入操作<br>...<br>插入操作完成<br>";
    return true;
}

?>

```

测试文件一、(ModelTest1.php) (查询班级标号(cls_id)为 2 的班级的学生的名单)

```

<?php
error_reporting(E_ALL);
require_once "Db.php";
require_once "Model.php";
require_once "Manager.php";

```

```
$classId = 2;

require_once "./Model/ClassManager.php";
$manager = new ClassModelManager();
$model = $manager->findOneModel($classId);
$data = &$model->getStudent();
foreach($data as $value)
echo "编号:".$value["stu_id"]." ----- 姓名: ".$value["stu_name"]."<br>";
?>
```

返回的结果是：

```
编号:1 ----- 姓名: student1
编号:2 ----- 姓名: student2
```

测试文件二、(ModelTest2.php) (查询班级标号(cls_id)为1的班级的学生的名单)

```
<?php
error_reporting(E_ALL);
require_once "Db.php";
require_once "Model.php";
require_once "Manager.php";

$classId = 1;

require_once "./Model/ClassManager.php";
$manager = new ClassModelManager();
$model = $manager->findOneModel($classId);
$data = &$model->getStudent();
foreach($data as $value)
echo "编号:".$value["stu_id"]." ----- 姓名: ".$value["stu_name"]."<br>";
?>
```

返回的结果是：

```
该班级不存在！
```

测试文件三、(ModelTest3.php) (执行数据库的插入工作, 向 student 表添加数据)

```
<?php
error_reporting(E_ALL);
require_once "Db.php";
require_once "Model.php";
require_once "Manager.php";

$data = array("stu_id"=>3, "stu_clsid"=>2, "stu_name"=>"student3");
require_once "./Model/StudentModel.php";
$model = new StudentModel($data);
require_once "./Model/StudentManager.php";
$manager = new StudentManager($data);
$result = $manager->insert($model);
echo $result? "<h2>插入操作成功</h2>": "<h2>插入操作失败</h2>";
?>
```

返回的结果是：

```
正在进行插入操作
...
插入操作完成

插入操作成功
```

测试文件四、(ModelTest4.php) (执行数据库的插入工作, 向 student 表添加数据)

```
<?php
error_reporting(E_ALL);
require_once "Db.php";
require_once "Model.php";
require_once "Manager.php";

$data = array("stu_id"=>3, "stu_name"=>"student3");
require_once "./Model/StudentModel.php";
$model = new StudentModel($data);
require_once "./Model/StudentManager.php";
$manager = new StudentManager($data);
$result = $manager->insert($model);
echo $result? "<h2>插入操作成功</h2>": "<h2>插入操作失败</h2>";
?>
```

返回的结果是：

stu_clsid 的数值不能为空

结果分析：

StudentModel 中“match”的规定 stu_clsid 的值是不能为空的，而代码中代码中 \$data = array("stu_id"=>3, "stu_name"=>"student3"); 缺少 stu_clsid 的值，因此不能通过数据的完整性校验，报错。

用 PHP 开始你的 MVC (四) 实现 View 层

文章整理：朱英 文章来源：网络

MVC 模式的 view 层的主要任务是进行页面的和结果的显示工作，在 php 的实现过程中，主要是体现为一个模板（使用模板，可以达到 php 代码和 html 代码分离的目的，这样代码和页面的维护就方便多了，便于管理和页面的更换，可以真正的划分程序员、美工的分工）的解析过程：

首先， controller 层从 model 层得到数据

其次， controller 层将数据交给 view 层

再次， view 层的接口将数据按一定的方式传给模板解析类，

最后，模板解析类将数据解析到模板中，然后显示。

下面是一个具体的实现例子

目录结构

```
|- ClassRenderTest.php           //测试解析 classlist.html
|- StudentRenderTest.php         //测试解析 studentlist.html
|- render / TemplateParser.php   //模板解析类
|- render / Render.php          //解析模板的所有类的基础类
|- render / StudentRender.php    //解析模板 studentlist.html 的类
|- render / ClassRender.php      //解析模板 classlist.html 的类
|- template / studentlist.html  //模板文件
|- template / classlist.html    //模板文件
```

注意：

- 1、这里模板解析类选用了简单的“TemplateParser.php”，根据个人的需要你可以选用任何一种模板解析类；
- 2、如果每个模板解析都直接调用“TemplateParser.php”，可能会有大量的重复代码出现，这是 oo 思想所不允许出现的。因此采用“Render.php”对它进行包装，然后再对“Render.php”里面的 Render 类进行扩展，来对不同文件模板进行解析；
- 3、不同的模板解析类的，使用的方法是不同的，他们的包装方式也可能不同。
- 4、“StudentRender.php”“ClassRender.php”就是包装过的 Render 类，分别用来满足解析“studentlist.html”“classlist.html”的需要。

文件 1: classlist.html

```
current time is : _now_ <BR><BR>
current school class list :
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>NAME</TH>
        <TH>GRADE</TH>
        <TH>CLASS</TH>
    </TR>
    BEGIN_classlist_
    <TR>
        <TD>_cid_</TD>
        <TD>_cname_</TD>
        <TD>_grade_</TD>
        <TD>_class_</TD>
    </TR>
    END_classlist_
</TABLE>
```

文件 2: studentlist.html

```
current time is : _now_ <BR><BR>
current class is :
<TABLE border=1>
    BEGIN_classinfo_
    <TR>
        <TH>class id: _cid_</TH>
        <TH>class name: _cname_</TH>
        <TH>class grade: _grade_</TH>
        <TH>class num: _class_</TH>
    </TR>
    END_classinfo_
</TABLE><br>
current class's student :
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>NAME</TH>
        <TH>SCORE</TH>
    </TR>
    BEGIN_studentlist_
    <TR>
```

```
<TD>_sid_</TD>
<TD>_sname_</TD>
<TD>_score_</TD>
</TR>
END_studentlist_
</TABLE>
```

文件 3: TemplateParser.php

下面的模板解析类是笔者临时写的一个简单的模板解析类，功能很少，没有真正什么使用价值。但是在这里可以满足这篇文章讲解的需要。

同时，如果以前没有接触过模板解析，对模板解析的实现方法有一定疑问的同僚，可以研究一下这个简单类实现解析的方式，代码挺简单的应该能看懂的。

这个解析类有自己的模板结构，“块”（要进行循环显示的地方）的定义如下：

```
BEGIN_你的块名_
.....html 代码.....
_你的块里面的变量的名称_
.....html 代码.....
END_你的块名_
```

变量的定义方式如下：

```
.....html 代码.....
_你的块里面的变量的名称_
.....html 代码.....
```

具体的“块”和“变量”的使用参考上面的两个模板

```
<?php
/*
 * template parse class
 */
class TemplateParser{
    /*
     * @var template root directory
     */
    var $root = '';
    /*
     * @var template filename
     */
    var $tpl = '';
    /*
     * @var data for parse template
     */
```

```
/*
var $data = array();
/*
 * @var template parse result
 */
var $result = ''';
/*
 * construct function
 */
function TemplateParser($root='') {
    $this->root = $root;
}
/*
 * set template file name
 */
function loadTemplateFile($tplFile) {
    $this->tpl = $tplFile;
}
/*
 * set global var value;
 *
 * @param $varName  global var name
 * @param $data      var's value
 */
function setData($varName, $data) {
    $this->data['__ALL__'][$varName] = $data;
}
/*
 * set global var value;
 *
 * @param $blockName template block name
 * @param $data        var value
 * @param $rec         var value
 */
function setBlockData($blockName, &$data, $rec=false) {
$this->data[$blockName] = &$data;
    $this->rec[$blockName] = $rec;
}
/*
 * parse template action
 */
function parse() {
    $tplstr = file_get_contents("{$this->root}/{$this->tpl}");
    foreach($this->data as $block=>$value) {
```

```
$tag = "|BEGIN_{\$block}_(.*)END_{\$block}|sm";
preg_match($tag, $tplstr, $tmpdata);
if($tmpdata[1]!=null) {
    $tmpstr = '';
    if($this->rec[$block]) {
        foreach($value as $subValue)
            $tmpstr .= $this->_parseBlock($tmpdata[1],
$subValue);
    } else{
        $tmpstr .= $this->_parseBlock($tmpdata[1],
$value);
    }
}
$tplstr =
preg_replace("|BEGIN_{\$block}_(.*)END_{\$block}|sm", $tmpstr,
$tplstr);
}
$tplstr = $this->_parseBlock($tplstr,
$this->data['__ALL__']);
$this->result = $tplstr;
}
/*
 * parse block
 *
 * @param $str      string  one block string
 * @param $data     array   data for parse
 */
function _parseBlock($str, $data) {
    foreach($data as $key=>$value) {
        $keys[] = "_{$key}_";
        $values[] = "$value";
    }
    return str_replace($keys, $values, $str);
}
/*
 * return parse result
 */
function get() {
return $this->result;
}
/*
 * show parse result
 */
function show() {
```

```
echo $this->result;
    }
}
?>
```

文件 4: Render.php

```
<?php
/*
 * include template parser class
 */
require_once "TemplateParser.php";
/*
 * base render class
 */
class Render{
    /*
     * @var object of template parser class
    */
    var $parser = "";
    /*
     * instruct function
    */
    function Render($root, $tplFile) {
        $this->parser = new TemplateParser($root);
        $this->parser->loadTemplateFile($tplFile);
    }
    /*
     * add data to template parser
     *
     * @param $data array() data for parse
    */
    function initData(&$data) {
        return ;
    }
    /*
     * show template parse result
    */
    function show() {
        $this->parser->parse();
        $this->parser->show();
    }
}
?>
```

文件 5: StudentRender.php

```
<?php
/*
 * include base class
 */
require_once "Render.php";
/*
 * shudent list show render
 */
class StudentRender extends Render{

    function StudentRender() {
        parent::Render('./template', 'studentlist.html');
    }

    function initData($data) {
        $this->parser->setData('now', date('Y-m-d H:i:s'));
        $this->parser->setBlockData('classinfo', $data['class'], false);
        $this->parser->setBlockData('studentlist',
$data['student'], true);
    }
}
?>
```

文件 6: ClassRender.php

```
<?php
/*
 * include base class
 */
require_once "Render.php";
/*
 * class list show render
 */
class ClassRender extends Render{

    function ClassRender() {
        parent::Render('./template', 'classlist.html');
    }

    function initData(&$data) {
        $this->parser->setData('now', date('Y-m-d H:i:s'));
        $this->parser->setBlockData('classlist', &$data['class'],
true);
    }
}
```

```
?>
```

下面两个是测试文件，第一个比较简单一些

测试文件 1: ClassRenderTest.php

```
<?php
/*
 * include "class list show render"
 */
require_once "render/ClassRender.php";
/*
 * data for parse
 *
 * include (class list)
 */
$data = array('class' =>array('1' =>array('cid' =>1,
'cname' =>'class one', 'grade' =>3, 'class' =>1),
'2' =>array('cid' =>2, 'cname' =>'class
two', 'grade' =>3, 'class' =>2),
'3' =>array('cid' =>3,
'cname' =>'class three', 'grade' =>4, 'class' =>1),
'4' =>array('cid' =>4,
'cname' =>'class four', 'grade' =>4, 'class' =>2),
'5' =>array('cid' =>5,
'cname' =>'class five', 'grade' =>5, 'class' =>1));
/*
 * do template parse
 */
doRender($data);
/*
 * may see as controller's action, use to parse template
 */
function doRender(&$data) {
    $render = new ClassRender();
    $render->initData($data);
    $render->show();
}
?>
```

运行结果:

```
current time is : 2004-05-10 23:51:26 <BR><BR>
current school class list :
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>NAME</TH>
        <TH>GRADE</TH>
        <TH>CLASS</TH>
    </TR>
    <TR>
        <TD>1</TD>
        <TD>class one</TD>
        <TD>3</TD>
        <TD>1</TD>
    </TR>
    <TR>
        <TD>2</TD>
        <TD>class two</TD>
        <TD>3</TD>
        <TD>2</TD>
    </TR>
    <TR>
        <TD>3</TD>
        <TD>class three</TD>
        <TD>4</TD>
        <TD>1</TD>
    </TR>
    <TR>
        <TD>4</TD>
        <TD>class four</TD>
        <TD>4</TD>
        <TD>2</TD>
    </TR>
    <TR>
        <TD>5</TD>
        <TD>class five</TD>
        <TD>5</TD>
        <TD>1</TD>
    </TR>
</TABLE>
```

测试文件 2: StudentRenderTest.php

```
<?php
/*
```

```

* include "shudent list show render"
*/
require_once "render/StudentRender.php";
/*
 * data for parse
 *
 * include (one class info) and (one class's student list)
*/
$data = array('class' => array('cid' =>1, 'cname' =>'class one',
'grade' =>3, 'class' =>1),
    'student' =>array('1' =>array('sid' =>1, 'sname' =>'stu
one', 'score' =>100),
        '2' =>array('sid' =>2, 'sname' =>'stu
two', 'score' =>90),
            '3' =>array('sid' =>3,
'sname' =>'stu three', 'score' =>80),
            '4' =>array('sid' =>4,
'sname' =>'stu four', 'score' =>95),
            '5' =>array('sid' =>5,
'sname' =>'stu five', 'score' =>55));
/*
 * do template parse
*/
doRender($data);
/*
 * may see as controller's action, use to parse template
*/
function doRender(&$data) {
    $render = new StudentRender();
    $render->initData($data);
    $render->show();
}
?>

```

运行结果:

```

current time is : 2004-05-10 23:52:22 <BR><BR>
current class is :
<TABLE border=1>
<TR>
    <TH>class id: 1</TH>
    <TH>class name: class one</TH>
    <TH>class grade: 3</TH>
    <TH>class num: 1</TH>
</TR>

```

```
</TABLE><br>
current class's student :
<TABLE border=1>
    <TR>
        <TH>ID</TH>
        <TH>NAME</TH>
        <TH>SCORE</TH>
    </TR>
<TR>
    <TD>1</TD>
    <TD>stu one</TD>
    <TD>100</TD>
</TR>
<TR>
    <TD>2</TD>
    <TD>stu two</TD>
    <TD>90</TD>
</TR>
<TR>
    <TD>3</TD>
    <TD>stu three</TD>
    <TD>80</TD>
</TR>
<TR>
    <TD>4</TD>
    <TD>stu four</TD>
    <TD>95</TD>
</TR>
<TR>
    <TD>5</TD>
    <TD>stu five</TD>
    <TD>55</TD>
</TR>
</TABLE>
```