

主讲教师：传智.神龙教主

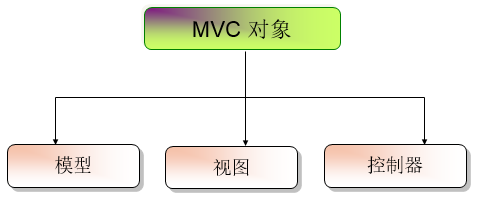
传智播客 PHP学院 <http://php.itcast.cn> 发布

## 回顾

1. clone：复制对象
2. \_\_clone()：当执行clone命令的时候自动调用的魔术方法
3. 单例模式：
   1. 一个类只能有一个对象
   2. 三私一公
      1. 私有的静态属性：保存当前类的实例
      2. 私有的构造方法：放在在类的外部实例化
      3. 私有的\_\_clone()：防止在类的外部克隆对象
      4. 公有的静态方法：用来获取当前对象的实例
4. 在项目层面的单例模式：将多个单例放到一个静态数组中
5. 工厂模式：传递不同参数获取不同的对象
6. 策略模式：传递不同参数调用不同的方法
7. 魔术方法：
   1. \_\_construct()：构造函数
   2. \_\_destruct()：析构函数
   3. \_\_autoload()：当调用不存在的类的时候自动调用
   4. \_\_tostring()：当将对象当成字符串使用的时候自动调用
   5. \_\_invoke()：当将对象当成函数使用的时候自动调用
   6. \_\_set()：当对无法访问的属性的赋值时候自动调用
   7. \_\_get()：当对获取无法访问的的属性值的时候自动调用
   8. \_\_unset()：当销毁无法访问的属性的时候自动调用
   9. \_\_isset()：当判断无法访问的属性是否存在的时候自动调用
   10. \_\_call()：当调用无法访问的普通方法时候自动触发
   11. \_\_callstatic()：当无法通过静态方式访问方法时自动触发
8. Iterator：(迭代器) 用来遍历对象中的数组类型的属性
   1. rewind()：指针复位
   2. valid()：判断指针是否合法
   3. current()：获取当前的数据
   4. key()：获取当前键
   5. next()：将指针下移一位
9. 数组(对象)序列号和反序列号
   1. serialize：序列化
   2. unserialize：反序列化
10. 通过单例模式实现数据库连接

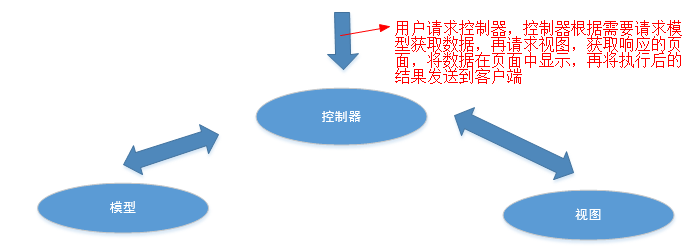
## MVC

1. MVC是一个设计思想。一个项目由三部分组成



* 1. M（Model）：模型层，用来操作数据的
  2. V（View）：视图层，用来显示数据
  3. C（Controller）：控制器层，用来执行业务逻辑的（调用数据并调用视图）

1. MVC 的优点
   1. 多人合作很方便
   2. 代码的可重用型非常好，（增强代码的健壮性）
   3. 项目的维护变得更加容易
2. MVC的请求示意图



## 演化一

获取商品的内容，在上一将的MySQLDB.class.php中，循环显示products表的数据

<?php

class MySQLDB {

private $host; //主机IP

private $port; //端口号

private $user; //用户名

private $pwd; //密码

private $charset; //字符编码

private $dbname; //连接的数据库

private $link; //保存数据库连接对象【可以省略】

private static $instance; //保存MySQLDB的实例

//初始化连接数据库参数

private function initParam($config) {

$this->host=isset($config['host'])?$config['host']:'';

$this->port=isset($config['port'])?$config['port']:'3306';

$this->user=isset($config['user'])?$config['user']:'';

$this->pwd=isset($config['pwd'])?$config['pwd']:'';

$this->charset=isset($config['charset'])?$config['charset']:'utf8';

$this->dbname=isset($config['dbname'])?$config['dbname']:'';

}

//连接数据库

private function connect() {

$this->link=@mysql\_connect("{$this->host}:{$this->port}",$this->user,$this->pwd) or die('数据库连接失败');

}

//设置字符编码

private function setCharset() {

$sql="set names '{$this->charset}'";

$this->query($sql);

}

//选择数据库

private function selectDB() {

$sql="use `{$this->dbname}`";

$this->query($sql);

}

/\*\*

\*此方法用来执行SQL语句

\*如果是数据查询语句，成功返回结果集，失败返回false

\*如果是数据操作语句，成功返回true,失败返回false;

\*/

public function query($sql) {

if(!$result=mysql\_query($sql,$this->link)){

echo 'SQL语句执行失败<br>';

echo '错误编号：'.mysql\_errno(),'<br>';

echo '错误信息：'.mysql\_error(),'<br>';

echo '错误的SQL语句',$sql,'<br>';

exit;

}

return $result;

}

/\*\*

\*私有的构造函数防止在类的外部实例化对象

\*@param $config array 连接数据库的参数

\*/

private function \_\_construct($config) {

$this->initParam($config);

$this->connect();

$this->setCharset();

$this->selectDB();

}

//私有的\_\_clone()用来阻止是类的外部克隆对象

private function \_\_clone() {

}

/\*\*

\*公有的静态方法用来获取MySQLDB类的实例

\*@return object MySQLDB的实例

\*/

public static function getInstance($config=array()) {

if(!self::$instance instanceof self)

self::$instance=new self($config);

return self::$instance;

}

/\*\*

\*从数据库获取所有数据

\*@param $sql string SQL语句

\*@param $fetch\_type string assoc|row|array

\*@return array 将表中的数据匹配成二维数组返回

\*/

public function fetchAll($sql,$fetch\_type='assoc') {

$rs=$this->query($sql);

$fetch\_types=array('assoc','row','array');

if(!in\_array($fetch\_type,$fetch\_types))

$fetch\_type='assoc';

$fetch\_fun='mysql\_fetch\_'.$fetch\_type;

$array=array();

while($rows=$fetch\_fun($rs)){

$array[]=$rows;

}

return $array;

}

/\*\*

\*获取记录的第一条记录

\*/

public function fetchRow($sql,$fetch\_type='assoc') {

$rs=$this->fetchAll($sql,$fetch\_type);

return empty($rs)?null:$rs[0];

}

/\*\*

\*获取记录的第一行第一列

\*/

public function fetchColumn($sql) {

$rs=$this->fetchRow($sql,'row');

return empty($rs)?null:$rs[0];

}

}

//获取数据

$config=array(

'host' => '127.0.0.1',

'user' => 'root',

'pwd' => 'aa',

'dbname'=> 'php4'

);

$db=MySQLDB::getInstance($config);

$rs=$db->fetchAll("select \* from products");

?>

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>无标题文档</title>

<style type="text/css">

table,td,th{

border:#000 solid 1px;

}

table{

width:780px;

margin:auto;

}

</style>

</head>

<body>

<table>

<tr>

<th>编号</th>

<th>商品名称</th>

<th>商品规格</th>

<th>价格</th>

<th>库存量</th>

<th>缩略图</th>

<th>网址</th>

</tr>

<?php foreach($rs as $rows):?>

<tr>

<td><?php echo $rows['proID']?></td>

<td><?php echo $rows['proname']?></td>

<td><?php echo $rows['proguige']?></td>

<td><?php echo $rows['proprice']?></td>

<td><?php echo $rows['proamount']?></td>

<td>

<?php

echo empty($rows['proimages'])?'图片暂缺':'<img src="'.$rows['proimages'].'">';

?>

</td>

<td><?php echo empty($rows['proweb'])?'网址暂缺':$rows['proweb']?></td>

</tr>

<?php endforeach;?>

</table>

</body>

</html>

## 演化二

1. 将MySQLDB类单独存放在MySQLDB.class.php页面中
2. 新建一个showList.html页面，将显示数据的代码放到此文件中

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>无标题文档</title>

<style type="text/css">

table,td,th{

border:#000 solid 1px;

}

table{

width:780px;

margin:auto;

}

</style>

</head>

<body>

<table>

<tr>

<th>编号</th>

<th>商品名称</th>

<th>商品规格</th>

<th>价格</th>

<th>库存量</th>

<th>缩略图</th>

<th>网址</th>

</tr>

<?php foreach($rs as $rows):?>

<tr>

<td><?php echo $rows['proID']?></td>

<td><?php echo $rows['proname']?></td>

<td><?php echo $rows['proguige']?></td>

<td><?php echo $rows['proprice']?></td>

<td><?php echo $rows['proamount']?></td>

<td>

<?php

echo empty($rows['proimages'])?'图片暂缺':'<img src="'.$rows['proimages'].'">';

?>

</td>

<td><?php echo empty($rows['proweb'])?'网址暂缺':$rows['proweb']?></td>

</tr>

<?php endforeach;?>

</table>

</body>

</html>

1. 新建一个index.php，用来获取数据并显示视图

<?php

//自动加载类

function \_\_autoload($class\_name) {

require "./{$class\_name}.class.php";

}

//获取数据

$config=array(

'host' => '127.0.0.1',

'user' => 'root',

'pwd' => 'aa',

'dbname'=> 'php4'

);

$db=MySQLDB::getInstance($config);

$rs=$db->fetchAll("select \* from products");

require './showList.html';

## 演化三

1. 一个表就是一个模型类
2. 模型类以Model结尾，比如Products表对应的模型ProductsModel
3. 类文件以.class.php结尾

将对Products表操作的代码封装到ProductsModel.class.php页面中

<?php

/\*\*

\*ProductsModel，用来操作Products表

\*/

class ProductsModel {

protected $db; //用来保存MySQLDB的单例

/\*\*

\*构造函数初始化连接数据库

\*/

public function \_\_construct() {

$this->initDB();

}

//获取MySQLDBm类的实例

private function initDB() {

$config=array(

'host' => '127.0.0.1',

'user' => 'root',

'pwd' => 'aa',

'dbname'=> 'php4'

);

$this->db=MySQLDB::getInstance($config);

}

/\*\*

\*获取Products表的所有商品

\*/

public function getList() {

return $this->db->fetchAll("select \* from products");

}

}

index.php页面更改如下：

<?php

//自动加载类

function \_\_autoload($class\_name) {

require "./{$class\_name}.class.php";

}

//获取数据

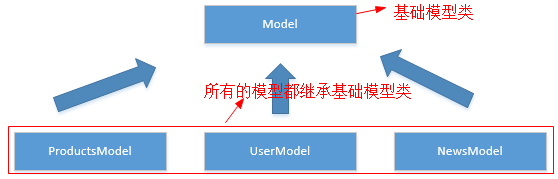
$model=new ProductsModel();

$rs=$model->getList();

require './showList.html';

## 演化四

每个表的模型都需要连接数据库，我们可以将连接数据库的代码封装到所有模型类的父类中（基础模型类）



基础模型类(Model.class.php)

<?php

/\*\*

\*基础模型类，用来封装所有模型的公共属性和方法

\*/

class Model {

protected $db; //用来保存MySQLDB的单例

/\*\*

\*构造函数初始化连接数据库

\*/

public function \_\_construct() {

$this->initDB();

}

//获取MySQLDBm类的实例

private function initDB() {

$config=array(

'host' => '127.0.0.1',

'user' => 'root',

'pwd' => 'aa',

'dbname'=> 'php4'

);

$this->db=MySQLDB::getInstance($config);

}

}

所有的模型都继承基础模型

class ProductsModel extends Model {

/\*\*

\*获取Products表的所有商品

\*/

public function getList() {

return $this->db->fetchAll("select \* from products");

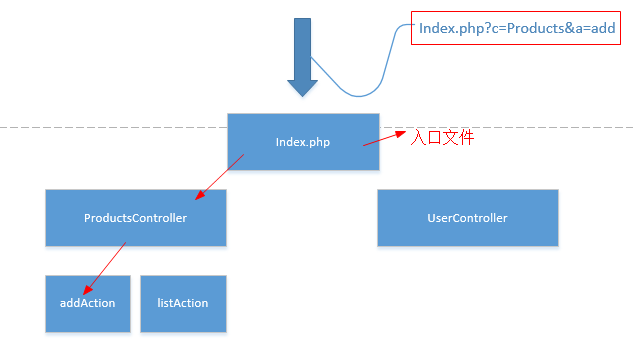
}

}

## 演化五

1. 一个模块就是一个控制器
2. 控制器必须以Controller结尾（控制器名+Controller）
3. 控制器中的方法名以Action结尾（方法名+Action）
4. 通过传递参数的方法请求分发

我们用c表示控制器,a表示方法



新建一个ProductsController.class.php页面

/\*\*

\*控制器必须以Controller结尾(控制器名+Controller)

\*方法名必须以Action结尾 （方法名+Action）

\*/

class ProductsController {

//获取商品的所有数据

public function listAction() {

$model=new ProductsModel();

$rs=$model->getList();

require './showList.html';

}

public function addAction() {

require './add.html';

}

public function modifyAction() {

require './modify.html';

}

public function delAction() {

echo 'del';

}

}

每个请求都要从index.php开始，所以index.php页面成为入口文件，作用：用来做请求分发和自动加载类（明天的课程继承封装请求分发和自动加载类）

<?php

//自动加载类

function \_\_autoload($class\_name) {

require "./{$class\_name}.class.php";

}

$c=isset($\_GET['c'])?$\_GET['c']:'Products'; //获取控制器

$a=isset($\_GET['a'])?$\_GET['a']:'list'; //获取方法

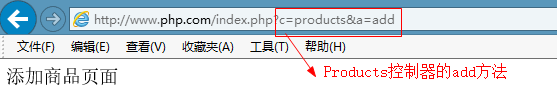
$controller\_name=$c.'Controller'; //拼接控制器名

$action\_name=$a.'Action'; //拼接方法名

$controller=new $controller\_name(); //实例化控制器

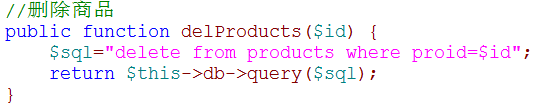
$controller->$action\_name(); //调用方法

### 请求控制器



## 删除商品

### 模型



### 视图

<input type="button" value="删除" onclick="if(confirm('确定要删除吗'))location.href='index.php?c=Products&a=del&id=<?php echo $rows['proID'];?>'">

### 控制器

