**第6章 面向对象**

**1. 理解面向对象的概念**

**A. 类和对象**

**B. 多态性**

**C. 继承**

**2. 在PHP中创建类、属性和操作**

**A. 类的结构**

**class classname{**

**}**

**class classname{**

**public $attribute1;**

**public $attribute2;**

**}**

**class classname{**

**function operation1(){**

**}**

**function operation2(){**

**}**

**}**

**B. 构造函数**

**class classname{**

**function \_\_construct($param){**

**echo 'Construction called with parameter ' . $param . '<br/>';**

**}**

**}**

**C. 析构函数**

**\_\_destruct()**

**3. 类的实例化**

**class classname{**

**function \_\_construct($param){**

**echo 'Construction called with parameter ' . $param . '<br/>';**

**}**

**}**

**$a = new classname('First');**

**$b = new classname('Second');**

**$c = new classname(1);**



**4. 使用类的属性**

**$this特殊指针：**

class classname{

public $attribute;

function operation($param){

$this->attribute = $param ;

echo $this->attribute;

}

}

$a = new classname();

$a->attribute = "value";

echo $a->attribute;

echo '<br/>';

$a->operation('var');

echo '<br/>';

echo $a->attribute;



**因为attribute的public权限，从类的外部可以直接访问类的属性了。**

**封装：**

class classname{

**private** $attribute;

function \_\_get($name){

return $this->$name;

}

function \_\_set($name,$value){

$this->$name = $value;

}

}

以上代码，为访问$attribute函数提供了最基本的功能。\_\_get() 函数返回了$attribute的值，而\_\_set()函数只是设置了$attribute的值。

**访问器函数：**

**\_\_get()函数带一个参数 ——属性名，并返回该属性的值**

**\_\_set()函数带两个参数——属性名和属性值。**

**我们并不会直接访问这些函数。\_\_开头，自动执行。工作原理：**

**实例化一个类：$a = new classname();**

**$a->attribute = 6; 该语句将间接调用\_\_set()函数，将$name参数的值设置为 attribute，而$value的值，被设置为5.必须编写\_\_set()函数来完成任何所需的错误检查。**

class classname{

private $attribute;

function \_\_get($name){

return $this->$name;

}

function \_\_set($name,$value){

$this->$name = $value;

}

}

$a = new classname();

$a->attribute = 1 ;

echo $a->attribute;



**$a->attribute ； 将间接调用\_\_get()函数，$name参数的值为 attribute。**

class classname{

**private** $attribute;

function **\_\_get**($name){

return $this->$name;

}

function **\_\_set**($name,$value){

**if(($name == "attribute") && ($value >=0) && ($value <=100))**

$this->$name = $value;

}

}

$a = new classname();

$a->attribute = **1056**;

var\_dump($a->attribute);



**5. 使用private和public关键字控制访问**

**默认为public。**

**6. 类操作的调用**

**class classname{**

**function operation1(){**

**}**

**function operation2($param1,$param2){**

**}**

**}**

**$a = new classname();**

**$a->operation1();**

**$a->operation2(12,"test");**

**如果操作具有返回值，可以捕获：**

**$x = $a->operation1();**

**$y = $a->operation2(12,"test");**

**7. 在PHP中实现继承**

**class A{**

**public $attribute1;**

**function operation1(){**

**}**

**}**

**class B extends A{**

**public $attribute2;**

**function operation2(){**

**}**

**}**

**则如下所有对于类B对象的操作和属性的访问，都是有效的：**

**$b = new B();**

**$b->operation1();**

**$b->attribute1 = 10;**

**$b->operation2();**

**$b->attribute2 = 30;**

**A. 通过继承使用private和protected访问修饰符控制可见性**

**如果一个属性或方法，被指定为private，它将不能被继承。**

**如果一个属性或方法，被指定为protected，它将在类外部不可见，可以被继承。**

**class A{**

**private function operation1(){**

**echo 'operation1 called';**

**}**

**protected function operation2(){**

**echo 'operation2 called';**

**}**

**public function operation3(){**

**echo 'operation3 called';**

**}**

**}**

**class B extends A{**

**function \_\_construct(){**

**$this->operation1(); //error!**

**$this->operation2();**

**$this->operation3();**

**}**

**}**

**$b = new B;**



**说明，私有操作，不能在子类中调用。**



**B. 重载**

**在子类中，再次声明相同的属性和操作，是有效的。**

**class A{**

**public $attribute = 'default value';**

**function operation(){**

**echo 'sth <br/>';**

**echo 'The value of $attribute is ' . $this->attribute .'<br/>';**

**}**

**}**

**class B extends A{**

**public $attribute = 'different value';**

**function operation(){**

**echo 'sth else <br/>';**

**echo 'The value of $attribute is ' . $this->attribute .'<br/>';**

**}**

**}**

**$a = new A;**

**$a->operation();**

**$b = new B;**

**$b->operation();**



**C. 使用final关键字禁止继承和重载**

**class A{**

**public $attribute = 'default value';**

**final function operation(){**

**echo 'sth <br/>';**

**echo 'The value of $attribute is ' . $this->attribute .'<br/>';**

**}**

**}**

**class B extends A{**

**public $attribute = 'different value';**

**function operation(){**

**echo 'sth else <br/>';**

**echo 'The value of $attribute is ' . $this->attribute .'<br/>';**

**}**

**}**

**$a = new A;**

**$a->operation();**

**$b = new B;**

**$b->operation();**



**注释红色代码，运行结果如下：**



**final class A{**

**...}**

**类A将不能被继承。**

**D. 理解多重继承**



**E. 实现接口**

**interface Displayable{**

**function diaplay();**

**}**

**class webPage implements Displayable{**

**function diaplay(){**

**#code...**

**}**

**}**

**$a = new webPage();**

**本类说明了多重继承的一种解决办法，因为webPage类可以继承一个类，同时又可以实现一个或多个接口。**

**8. 类的设计**

**web开发中可能使用的类包括网页、用户界面组件、购物车、错误处理、商品分类或顾客。**

**准备创建一个Page类，目的是减少创建一个新页面所需要的HTML代码。该类为建立新页面提供灵活的框架。**

**功能包括：**

**1）允许在需要修改某些页面元素的时候，只在一处进行修改。**

**2）页面大部分区域有默认内容，局部可以修改。**

**3）识别哪一个页面是当前浏览页面，并相应改变导航条。**

**4）允许使用特定的页面代替标准页面。**

**9. 编写类代码**

**类名 ： class Page{}**

**属性 ：public $content 页面的主要内容**

**public $title = 'TLA Consulting Pty Ltd'; 页面标题**

**public $keywords = 'TLA Consulting, Three Letter Abbreviation, some of my best friends are search engines';**

**导航条，按钮个数可能不同页面有所不同，因此使用一个数组，来保存按钮的文本标签以及该按钮指向的URL:**

**public $bottons = array("Home"=>"home.php", "Contact"=>"contact.php, "Services"=>"services.php",**

**"Site Map"=>"map.php");**

**类的操作：**

**public function \_\_set($name,$value){**

**$this->$name = $value;**

**}**

**显示页面，需要一个Display()函数：**

**public function Display(){**

**echo '<html>\n<head>\n';**

**$this->DisplayTitle(); //显示标题**

**$this->DisplayKeywords();//显示关键字**

**$this->DisplayStyles();//显示样式**

**echo "</head>\n<body>\n";**

**$this->DisplayHeader();//显示页顶**

**$this->DisplayMenu($this->buttons);//显示导航条**

**echo $this->content;**

**$this->DisplayFooter();//显示页脚**

**echo "</body>\n</html>\n";**

**page.inc程序清单如下：**

**—————————————————————————————————**

<?php

class Page

{

// class Page's attributes

public $content;

public $title = "TLA Consulting Pty Ltd";

public $keywords = "TLA Consulting, Three Letter Abbreviation,

some of my best friends are search engines";

public $buttons = array("Home" => "home.php",

"Contact" => "contact.php",

"Services" => "services.php",

"Site Map" => "map.php"

);

// class Page's operations

public function \_\_set($name, $value)

{

$this->$name = $value;

}

public function Display()

{

echo "<html>\n<head>\n";

$this -> DisplayTitle();

$this -> DisplayKeywords();

$this -> DisplayStyles();

echo "</head>\n<body>\n";

$this -> DisplayHeader();

$this -> DisplayMenu($this->buttons);

echo $this->content;

$this -> DisplayFooter();

echo "</body>\n</html>\n";

}

public function DisplayTitle()

{

echo "<title>".$this->title."</title>";

}

public function DisplayKeywords()

{

echo "<meta name=\"keywords\"

content=\"".$this->keywords."\"/>";

}

public function DisplayStyles()

{

?>

<style>

h1 {

color:white; font-size:24pt; text-align:center;

font-family:arial,sans-serif

}

.menu {

color:white; font-size:12pt; text-align:center;

font-family:arial,sans-serif; font-weight:bold

}

td {

background:black

}

p {

color:black; font-size:12pt; text-align:justify;

font-family:arial,sans-serif

}

p.foot {

color:white; font-size:9pt; text-align:center;

font-family:arial,sans-serif; font-weight:bold

}

a:link,a:visited,a:active {

color:white

}

</style>

<?php

}

public function DisplayHeader()

{

?>

<table width="100%" cellpadding="12"

cellspacing="0" border="0">

<tr bgcolor ="black">

<td align ="left"><img src = "logo.gif" /></td>

<td>

<h1>TLA Consulting Pty Ltd</h1>

</td>

<td align ="right"><img src = "logo.gif" /></td>

</tr>

</table>

<?php

}

public function DisplayMenu($buttons)

{

echo "<table width=\"100%\" bgcolor=\"white\"

cellpadding=\"4\" cellspacing=\"4\">\n";

echo "<tr>\n";

//calculate button size

$width = 100/count($buttons);

while (list($name, $url) = each($buttons)) {

$this -> DisplayButton($width, $name, $url,

!$this->IsURLCurrentPage($url));

}

echo "</tr>\n";

echo "</table>\n";

}

public function IsURLCurrentPage($url)

{

if(strpos($\_SERVER['PHP\_SELF'], $url )==false)

{

return false;

}

else

{

return true;

}

}

public function DisplayButton($width,$name,$url,$active = true)

{

if ($active) {

echo "<td width = \"".$width."%\">

<a href=\"".$url."\">

<img src=\"s-logo.gif\" alt=\"".$name."\" border=\"0\" /></a>

<a href=\"".$url."\"><span class=\"menu\">".$name."</span></a>

</td>";

} else {

echo "<td width=\"".$width."%\">

<img src=\"side-logo.gif\">

<span class=\"menu\">".$name."</span>

</td>";

}

}

public function DisplayFooter()

{

?>

<table width="100%" bgcolor="black" cellpadding="12" border="0">

<tr>

<td>

<p class="foot">&copy; TLA Consulting Pty Ltd.</p>

<p class="foot">Please see our <a href ="">legal

information page</a></p>

</td>

</tr>

</table>

<?php

}

}

?>

————————————————————————————————————————

**home.php——首页使用Page类完成生成页面内容的大部分工作**

**—————————————————————————————————**

**<?php**

**require("page.inc");**

**$homepage = new Page();**

**$homepage->content ="<p>Welcome to the home of TLA Consulting.**

**Please take some time to get to know us.</p>**

**<p>We specialize in serving your business needs**

**and hope to hear from you soon.</p>";**

**$homepage->Display();**

**?>**

**—————————————————————————————————**



**如果希望网站的一些地方使用不同的标准页面，可以用继承来创建新类。**

**services.php——services页面继承了Page类但是重载了Display()操作，从而改变了其输出结果。**

**—————————————————————————————————**

**<?php**

**require ("page.inc");**

**class ServicesPage extends Page**

**{**

**private $row2buttons = array(**

**"Re-engineering" => "reengineering.php",**

**"Standards Compliance" => "standards.php",**

**"Buzzword Compliance" => "buzzword.php",**

**"Mission Statements" => "mission.php"**

**);**

**public function Display()**

**{**

**echo "<html>\n<head>\n";**

**$this -> DisplayTitle();**

**$this -> DisplayKeywords();**

**$this -> DisplayStyles();**

**echo "</head>\n<body>\n";**

**$this -> DisplayHeader();**

**$this -> DisplayMenu($this->buttons);**

**$this -> DisplayMenu($this->row2buttons);**

**echo $this->content;**

**$this -> DisplayFooter();**

**echo "</body>\n</html>\n";**

**}**

**}**

**$services = new ServicesPage();**

**$services -> content ="<p>At TLA Consulting, we offer a number**

**of services. Perhaps the productivity of your employees would**

**improve if we re-engineered your business. Maybe all your business**

**needs is a fresh mission statement, or a new batch of**

**buzzwords.</p>";**

**$services -> Display();**

**?>**

**—————————————————————————————————**

**比原函数Display()多了一行：**

**$this -> DisplayMenu($this->row2buttons);**

**第二次调用，再创建一个菜单条。**



**10. 理解PHP面向对象的高级功能**

**A. 使用Per—Class常量**

**class Math{**

**const pi = 3.14159;**

**}**

**echo 'Math::pi = ' . Math::pi . '\n';**

**B. 实现静态方法**

**class Math{**

**static function squared($input){**

**return $input\*$input;**

**}**

**}**

**echo Math::squared(8);**

**在未初始化该类的情况下调用这个方法。**

**注意：在一个静态方法中，不能使用this关键字，因为可能会没有可以引用的对象实例。**

**C. 检查类的类型和类型提示**

**在前面作为类A的子类而实现的类B例子中，如下语句：**

**{$b instanceof B}将返回true**

**{$b instanceof A}将返回true**

**{$b instanceof Displayable}将返回false**

**D. 延迟静态绑定**

**class A{**

**public static function who(){**

**echo \_\_CLASS\_\_;**

**}**

**public static function test(){**

**static::who(); //延迟绑定**

**}**

**}**

**class B extends A {**

**public static function who(){**

**echo \_\_CLASS\_\_;**

**}**

**}**

**B::test();**

**输出结果** 

**test() 方法中，static改为self，则输出A。**

**E. 克隆对象**

**$c = clone $b;**

**将创建与对象$b具有相同类的副本，而且具有相同的属性值。**

**F. 使用抽象类**

**abstract operation($param1,$param2);**

**包含抽象方法的任何类，自身必须是抽象的。**

**abstract class A{**

**abstract function operationX($param1,$param2);**

**}**

**G. 使用\_\_call()重载方法**

**\_\_get() \_\_set() \_\_construct() \_\_destruct() \_\_call()**

**这些双下划线开头的方法，是一些具有特殊意义的类方法，自动调用。**

**\_\_call()方法，须有两个参数：方法名和参数数组。**