iranphp articles

عنوان مقاله :	شیی گرایی در PHP بخش نخست
نگارنده :	هوتن اقاسپور
آدرس پست الکترونیک :	houtanal@yahoo.com
تاریخ نگارش :	

شیی گرایی در PHP بخش نخست :

Phpیک زبان شی گرای کامل نیست اما امکانات شی گرایی زیادی دارد.که در نسخه پنجم آن بهتر نیز شده است .

در این مقاله فرض من بر این است که شما زبان php تا حدودی بلدید .

شی گرایی زبان های برنامه نویسی را به دنیای واقعی نزدیک کرده است یعنی شما در برنامه نویسی شی گرا اشیایی را می سازید که توانایی هایی را دارند .

حالا کمی جلوتر برویم من یک دوست دارم که در برنامه نویسی شی گرا او را کلاس در نظر می گیریم.این دوست من اخلاق هایی دارد مثلا اگر به او فلان چیز را بگویید عصبانی خواهد شد این اخلاق ها را متد های کلاس می گوییم که در مقابل عمل عکس العمل نشان می دهند.یا کاری را انجام خواهند داد .

تعریف کلاس :

در php کلاس ها را به این گونه تعریف می کنیم :

```
<?php
Class classname{

}


//Example:

Class a{

//class codes

}
?>
```

خب حالا یک کلاس داریم.و در آن چند متغییر و تابع تعریف می کنیم .

```
<?php
Class a{
   var $i="hello,this is I variable";
   function a1(){
     print "hi,this is function a1()";
  }

}?>
```

حالا وقت استفاده از آن هاست شما دو راه معمول دارید یکی اینکه با ساختن یک شی جدید از کلاس از این به بعد از آن استفاده کنید یا مستقیما اجزلی کلاس را صدا بزنید .

راه اول :

از کلمه کلیدی new استفاده می کنیم.به مثال توجه کنید:

```
<?php
Class a{

function a1(){
     print "hi,this is function a1()";
  }
}

$j=new a();

$j->a1();
```

```
//This code would print on screen:hi,this is function a1()

?>
```

کلمه کلیدی new یک شی جدید از کلاس معرفی شده می سازد یعنی مثل این است که کلاس را درون شی کپی میکند (این مثال چندان هم علمی نیست اما برای فهمیدن بد نیست).پس از آن ما با عملگر -> به متد های کلاس دسترسی داریم .

راه دوم :

از عملگر:: استفاده می کنیم که مستقیما نام کلاس و سپس نام متد را می آوریم .

```
<?php
Class a{

function a1(){

print "hi,this is function a1()";

}

}

a::a1();

//This code would print on screen:hi,this is function a1()

?>
```

شما باید در هنگام کار بنا به نیاز خود از این عملگر ها استفاده کنید ولی ساختن یک شی معمول تر است زیرا در خواندن و فهم کد را راحت تر می کند و در هنگام گسترش پروژه نیز مشکل شما را کم خواهد کرد .

استفاده از $this

با $this ما به شی اجازه بازگشت به خود را می دهیم.یعنی ما می توانیم به اجزا کلاس جاری دسترسی پیدا کرده و آن ها را تغییر دهیم .

```
<?php

class name{

var $firstname="houtan";

var $lastname="alghaspour";

function print_name(){

Print "My First Name is: ".$this->firstname."<br>";

Print "My Last Name is: ".$this->lastname;

}

function change_name(){

$this->firstname="changed first name";

$this->lastname="changed last name";
```

```
}
}
$i=new name();
$i->print_name();
//change name
$i->change_name();
echo "<p>";
$i->print_name();
?>
```

خب شما اکنون می توانید یک کلاس تعریف کرده و از آن استفاده کنید.پس جلوتر می رویم .

ایجاد سازنده (Constractor)

سازنده متدی است که با هر بار ساختهش شدن یک شی از کلاس به طور خورکار اجرا خواهد شد.تعریف آن هم بینهایت ساده است!متدی تعریف کنید اسمش با اسم کلاس دقیقا یکی باشد

```
<?php
class a{
function a(){
print "You now create a new object";
}
}
$i=new a();
?>
```

ایجاد Destructor

کارDestructor دقیقا بر عکس سازند است.در php مانند زبانی مثل C# دقیقا Destructor به همان سادگی وجود ندارد اما میتوان بوسیله تابع register_shutdown_function() این کار را انجام داد

کار این تابع این است که در هنگام اتمام کد یک تابع را که برایش تعریف کرده اید اجرا میکند .

```
<?php
class a{
function a(){
print "Hello world!<p>";
}
function a2(){
```

```
print "destructoring...";

}

}

$i=new a();

register_shutdown_function($i->a2());

?>
```

وراثت (Inheritance)

از قدیم گفته اند "پسر کو ندارد نشان از پدر تو بیگانه خوانش نه پسر"!با استفاده از وراثت ما کلاسی تعریف می کنیم که می تواند از اجزای کلاس والد خود استفاده کند(متغییر ها و متد ها).اما عکس این مطلب صادق نیست .

برای اینکه بگوییم کلاس از کلاس دیگری ارث برده از کلمه کیدی **extends** استفاده می کنیم.(برنامه نویسان جاوا به خوبی این لغت را می شناسند)

```
<?php
//Example:

class a{

function a(){

print "This class a constructor<p>";

}

function a1(){

print "This function a1() from class a<p>";

}

}

class b extends a{

function b1(){

print "This is function b1() from class b";

}

}

  $i=new b();

  $i->a1();

  $i->b1();

?>
```

به یکی دیگرتوجه کنید :

```
<?php
```

```
class a{
   var $name="houtan";
   function a1(){
   print "<p>hello world!";
   }
}
class b extends a{
  function b(){
  $this->name="ali";
   print $this->name;

   }
}
  $i=new b();
  $i->a1();

?>
```

متغییر های عمومی و خصوصی (Private an public)

در زبان هایی مثل C# / C++ / Java شما به سادگی می توانید توابع و متغییر های خود را private و یا Public تعریف کنید از آنجا که php مستقیما روشی برای ایم کار ندارد بین برنامه نویسیا این زبان رسم شده تا از یک کاراکتر _ برای داده های خصوصی (Private) استفاده کنند. مثل نام گذاری لهستانی در C++ . دلیل این کار سادگی فهم کد برای سایرین و یا خود شما پس از چند ماه است

(به نقل از کتاب PHP developer's cookbook صفحه۲۰۹)

اما در مورد تعریف متغییر های عمومی می توانید از لغت کلیدی global استفاده کنید .

```
<?php
Function name()
{
global $myname;
$myname='houtan';
}
name();
echo $myname;
```

```
?>
```

توجه کنید اگر در خط ۷ تابع `name()` را حذف کنید هیچ چیز روی صفحه نمایش نشان داده نمی شود چون این تابع باید حداقل یک بار اجرا شود تا متغییر `myname` به صورت عمومی تعریف شود .
شما باید `global` را قبل از متغییری که می خواهید از آن استفاده کنید بگذارید .

```
<?php
Function name()
{
global $myname;
$myname='houtan';
}
name();
echo $myname."<p>";
function secondname(){
global $myname;
echo $myname;
}
secondname();
?>
```