

iranphp articles

عنوان مقاله :	کلاسها در PHP 4
نگارنده :	سید حمید رضا هاشمی گلپایگانی
آدرس پست الکترونیک :	
تاریخ نگارش :	

کلاسها در PHP چهار:

یکی از مسائلی که در PHP فهمیدن آن خیلی راحت نیست "مزیت استفاده از کلاسها" در برنامه نویسی می باشد. من قبلا با هیچ بانک اطلاعاتی در PHP کار نکرده بودم اما بعد از کمی تلاش دیدم که یادگیری دستورات مقدماتی آن چقدر راحت است اما در مقابل برنامه نویسی (OOP برنامه نویسی شیء گرا) نیز تا بحال انجام نداده بودم ولی پی بردم که فهمیدن مفاهیم و چرایی استفاده از آن بسیار مشکل بود. من فکر می کردم که برنامه نویسی شیء گرا حتما خیلی قدرتمند است و مزایای زیادی دارد اما از اونجائی که سعی می کردم قبل از تجربه هر چیز خودم را قانع کنم که اون چیز به دردم می خوره هیچ موقع دنبال اون نبودم.

تا اینکه چند روز پیش هنگامی که داشتم یک Function ساده می نوشتم و اون رو تغییر می دادم به ذهنم رسید که ببینم می شود این کار را توسط یک (Object شیء) انجام داد یا نه. سعی می کنم نتایجی رو که در هنگام این کار بهش رسیدم رو براتون به زبان ساده شرح بدم (Class کلاس) در حقیقت تعدادی از (Variable متغیرها) به همراه تعدادی (Function عملگر) است که این عملگرها بر روی آن متغیرها کار می کنند. اونها در حقیقت مفهوم یک چیز در دنیای واقعی هستند. به عبارت دیگر کلاسها یک شیء رو معرفی می کنند. نشانه یک کلاس زندگی واقعی و یا نفس کشیدن است که زیر ساختهای اون کلاس هستند.

فرض کنید که ما می خواهیم یک دوچرخه را شرح دهیم. کلاس مرتبط با یک دوچرخه می بایست متغیرهای زیر را داشته باشد:

```
$pedals, $chain, $front_wheel, $rear_wheel, $brakes, $handleBars
```

(بدنه نگهدارنده، ترمزها، چرخ عقب، چرخ جلو، زنجیر، پدالها)

عملگرهایی که یک دوچرخه دارد از قرار زیر است:

```
Stop(), Accelerate(), Coast(), TurnLeft(), TurnRight()
```

(ایستادن، شتاب دادن، راندن، پیچیدن به سمت راست، پیچیدن به سمت چپ)

شما هم اکنون می توانید فرض کنید کنید که توسط این اجزاء این دوچرخه می تواند حرکت کند و در حقیقت معنای دوچرخه بدهد. مثلا عملگر Accelerate می تواند متغیری به عنوان \$Braking_Force به عنوان ورودی دریافت کند و مثلا از متغیرهای \$brakes و \$wheels هم استفاده کند و مثلا نتیجه مطلب را به عنوان مقدار بازگشتی به برنامه برگرداند.

بسیار جالب به نظر می رسد. اما به نظر شما نمی شه که همه اینها را به صورت منظم توسط تعدادی متغیر و عملگر انجام داد؟ مسلما این کار امکان پذیر هستش. اما در صورتیکه شما فقط یک دوچرخه را بخواهید در برنامه خودتون استفاده کنید فرقی به حال شما ندارد که از کلاس استفاده کنید یا نه. اما اگر از تعدادی از دوچرخه ها بخواهید در برنامه استفاده کنید چطور؟ در اون صورت تعقیب اینکه هر دوچرخه هم اکنون در چه مرحله ای است و چه متغیری از آن باید به کدام عملگر فرستاده شود توسط برنامه نویس بسیار سخت و پیچیده می شود. اما در کلاس دیگر این مشکلات وجود ندارد و هر متغیر برای هر دوچرخه در اختیار عملگرهای آن است و هر دوچرخه به صورت جدا متغیرهایش برای عملگرها فرستاده می شود بدون اینکه شما نیازی به کنترل آنها داشته باشید همچنین استفاده از کلاس در برنامه های مختلف راحت است چرا که شما با یک کلاس آماده که قبلا در جای دیگر کار می کرده به راحتی می توانید کار کنید و نیازی به تغییر آن ندارید.

حالا اجازه بدید که یک مثال کاربردی و واقعی بزنیم که من در خیلی از صفحات وبی که می سازم از آن استفاده می کنم و ممکن است که بدرد شما هم بخورد. شما رو نمی دونم اما من موقعی که می خوام صفحات وب پویا رو برنامه نویسی کنم خیلی بدم میاد که همش به فکر روند منطقی درست بودن خروجی HTML برنامه باشم. برای همین هیچ موقع صفحات زیبایی در خروجی برنامه های من نیست چون از رنگها و فونتهای مختلف استفاده نمی کنم.

راه حل موجود استفاده از یک کلاس PHP برای تنظیم خروجی های HTML است. من این کلاس رو Style نامگذاری می کنم. این کلاس شامل متغیرهای زیر است که مهمترین تنظیمات HTML خروجی از برنامه می باشند:

```
<?php
class Style {
    var $text;
    var $alink;
    var $vlink;
```

```
var $link;  
var $bgcol;  
var $face;  
var $size;  
var $align;  
var $valign;  
}  
?>
```

مطمئن هستیم که شما با HTML آشنا هستید و این متغیرها برای شما آشنا هستند. هم اکنون سعی می کنیم که عملگری به نام **Style** برای تعیین نوع خروجی درست کنیم:

```
<?php  
class Style {  
function Style ($text="#000000",$alink="#AA00AA",  
    $vlink="#AA00AA",$link="#3333FF",  
    $bgcol="#999999",$face="Book Antiqua",$size=3,$align="CENTER",$valign="TOP")  
{  
$this->text=$text;  
$this->alink=$alink;  
$this->vlink=$vlink;  
$this->link=$link;  
$this->bgcol=$bgcol;  
$this->face=$face;  
$this->size=$size;  
$this->align=$align;  
$this->valign=$valign;  
}  
}  
?>
```

هنگامی که شما عملگری همانام با کلاس خودتون تعریف می کنید، این عملگر همزمان با تعریف شیء ای از نوع این کلاس اجرا خواهد شد. این عملگر، عملگر **constructor** یا سازنده نام می گیرد. این عملگر به شما این امکان را می دهد که در هنگام ایجاد شیء، برای همه متغیرها یک مقدار اولیه داشته باشید

```
<?php  
$Basic = new Style;  
?>
```

در اینجا شما شیء ای با نام **\$basic** توسط دستور **new** از نوع کلاس **Style** درست کرده اید.

همچنین این امکان وجود دارد در هنگامیکه می خواهید یک شیء جدید توسط یک کلاس ایجاد کنید مقدار اولیه متغیرهای آن را نیز در هنگام ایجاد خوتان انتخاب کنید. اما در صورتیکه مقدار یکی از آنها را شما بخواهید در هنگام ایجاد انتخاب کنید باید همه متغیرهای دیگر را نیز خودتان مقدار اولیه هایشان را تعیین کنید و این امکان وجود ندارد که فقط یکی یا چند تا رو شما تعیین کنید و بقیه رو مقدار اولیه مقدار دهی کند. به همین خاطر راه بهتری را برای این مشکل پیشنهاد می کنیم و آن استفاده از یک عملگر به نام **Set** در کلاس است که مقدار یک متغیر را در کلاس تغییر می دهد:

```
<?php  
Function Set($varname,$value) {  
$this->$varname=$value;  
}  
?>
```

بنابراین برای تغییر مقدار هر یک از متغیرها می توانیم از نمونه زیر استفاده کنیم:

```
<?php  
$Basic->Set('size','2');  
?>
```

شما می توانید از علامت **->** برای اشاره به متغیر و یا عملگر یک شیء استفاده کنید. بنابراین خط بالا به اجرا کننده برنامه می گوید که عملگر **Set** مربوط به شیء **\$Basic** را اجرا کن. از آنجایی که ما **\$Basic** را قبلا از نوع **Style** تعریف کردیم بنابراین عملگر **Set** این کلاس اجرا می شود و پارامترهای ورودی شما را می گیرد و برای شیء مورد درخواست تغییرات لازم را اجرا می کند. همچنین شما می توانید این کار را برای متغیرهای دیگر مانند **\$Basic->text** نیز انجام دهید و مقدار اولیه آنرا تغییر دهید. حالا می خواهیم برای عناوین جدولها **Style** جدیدی تعریف کنیم با یک سری تعریفهای متفاوت:

```
<?php
$header= new Style;
$header->Set('text', '#0000FF');
$header->Set('bgcol', '#000000');
?>
```

اینجا خیلی جالب شد در اینجا عناوین جدول من نوشته هایش به رنگ آبی و زمینه آن رنگ سیاه خواهد بود. همچنین من می خواهم نوشته های درون جدول کمی واضح تر از نوشته های دیگر در صفحه باشد. سیاه رنگ خوبی برای این نوشته ها است اما شاید بخواهم که نوشته های درون جدول کمی ریزتر باشند:

```
<?php
$tbody=new Style;
$tbody->Set('bgcol', '#AAAAAA');
$tbody->Set('size', 2);
?>
```

بسیار عالی خوب حالا با اینها چکار کنیم؟ خوشحالم که این سؤال براتون پیش اومد. در اینجا لازم است که یک سری عملگرهای جدید تعریف کنیم که کارها رو انجام بدهند. اولین چیزی که من می خواهم درست شود، تعریف بدنه اصلی می باشد. بنابراین من این کار رو می کنم

```
<?php
function Body() {
print "<BODY BGCOLOR=\"$this->bgcol\" ".
"TEXT=\"$this->text\" ".
"LINK=\"$this->link\" VLINK=\"$this->vlink\" ".
"ALINK=\"$this->alink"><FONT ".
"FACE=\"$this->face\" SIZE=$this->size>\n";
}
?>
```

این عملگر بدنه اصلی را مشخص می کند. همچنین اینجا استفاده از متغیر **\$this** برای ما مشخص شد. هنگامی که شما از این متغیر درون یک عملگر کلاس استفاده می کنید به اجرا کننده می فهمانید که منظور شما اشاره به متغیرهای همین شیء دارید. به عبارت دیگر به اجرا کننده می گوئید که بجای متغیر **\$this** نام همین شیء را قرار بده

مثلا هنگامی که شیء مورد اشاره شما **\$Basic** باشد متغیر **\$this->face** خواهد بود **\$Basic->face** که قبلا توضیح داده شد اگر دقت کنید می بینید که ما کاری را داریم انجام می دهیم که توسط عملگرهای عادی به این راحتی ها قابل انجام نیست. چرا که در عملگرهای عادی و خارج از کلاس شما باید پارامترهای ورودی برای هر بار اجرای عملگر را وارد کنید در حالی که اینجا هر بار اجرای عملگر نیازی به مشخص ساختن پارامترها وجود ندارد و خود اجرا کننده از آنها اطلاع دارد.

حالا فرض می کنیم که شما کلاس **Style** را در برنامه خود فراخوانی کرده اید و اشیاء مورد نیاز خود را ایجاد کرده اید و تغییرات مورد نیاز آنها را انجام داده اید و **<HTML>** و **<HEAD>/HEAD>** را نیز به خروجی فرستاده ایم. دستور زیر را استفاده می کنیم:

```
<?php
$Basic->Body();
?>
```

هم اکنون همه چیز آماده است که شما چک چیزی را در خروجی چاپ کنید. شما می توانید این کار را توسط دستورات معمول PHP انجام دهید. اما من این کار را نیز در اینجا با تعریف یک عملگر دیگر درون کلاس انجام می دهم. عملگر مورد اشاره به شرح زیر است:

```
<?php
function TextOut($message=" ") {
    print "
        <font face=".$this->face." size=".$this->size."
            color=".$this->text.">$message</font>
    ";
}
?>
```

این عملگر پیغامی را به عنوان پارامتر دریافت کرده و مطابق Style انتخاب شده در خروجی چاپ می کند. به همین صورت می توانیم پیغامهای زیر را چاپ کنیم:

```
<?php
$Basic->TextOut('This is my test message');
$Tbody->TextOut(' -- kinda neat, huh?');
?>
```

دقت کنید که چون هیچ در عملگر تعریف شده ما وجود ندارد برای همین هر دوی این پیغامها در یک خط نوشته خواهند شد با این تفاوت که پیغام دوم ریزتر از اولی خواهد بود چراکه از نوع \$Tbody می باشد. تفاوت دیگر در این دو پیغام زمینه های آنها می باشد که هنگام تعریف دو شیء \$Basic و \$Tbody آنها را شرح دادیم. تنها نکته ای که می ماند این است که در عملگر TextOut که تعریف کردیم یک مقدار اولیه برای \$message که پارامتر ورودی این عملگر است در نظر گرفتیم و آن " " است. کسانی که با HTML آشنا هستند می دانند که این مقدار یک فاصله-Non Breaking را در خروجی قرار می دهد و این برای این است که در صورتیکه این عملگر بدون پارامتر فراخوانی شد این را در خروجی قرار می دهد که خیلی جاها بدرد می خورد.

خوب در اینجا مسائل مربوط به درست کردن متنی که در آن کلمه هایی با Font ها و یا رنگ ها و ... دیگر باشد را حل کردیم. اما هنوز همه چیز تموم نشده است. مثلا اگر عملگر جدید به صورت زیر تعریف کنیم:

```
<?php
function TDOut ($message="&nbsp;", $colspan=1) {
    print "<TD COLSPAN=".$colspan." BGCOLOR=".$this->bgcol."
        ALIGN=".$this->align." VALIGN=".$this->valign.">";
    $this->TextOut($message);
    print "</TD>\n";
}
?>
```

حالا یک مساله جدید است. دقت کنید که می خواهیم در هر سطر و یا ستون جدول خود زمینه های مختلفی داشته باشیم. بنابراین این کار را می توانیم انجام دهیم:

```
<TABLE>
<TR>
<?php
$Theader->TDOut("Name",2);
$Theader->TDOut("Location",3);
?>
</TR>
<TR>
<?php
$Theader->TDOut("Last");
$Theader->TDOut("First");
$Theader->TDOut("City");
$Theader->TDOut("State/Province");
$Theader->TDOut("Country");
```

```
?>
</TR>
```

در اینجا می توانید ببینید که متغیر `colspan` چگونه کار می کند. در صورتیکه مقدار آنرا مشخص نکنید به صورت خودکار مقدار آن ۱ فرض خواهد شد همانگونه که می بینید. همچنین می بینید که در اینجا در سطر اول عنوان `Name` دارای ۲ ستون است و عنوان `Location` دارای ۳ ستون است. در سطر بعدی هر کدام دارای یک ستون می باشند همانطور که می بینید. حالا می خواهیم مقادیر جدول را پر کنیم:

```
<TR>
<?php
$tbody->TDOut("Kreislser");
$tbody->TDOut("Rod");
$tbody->TDOut("Cortlandt");
$tbody->TDOut("New York");
$tbody->TDOut("USA\"); ?>
</TR>
```

اما همانطور که می بینید این کار هنوز کمی طویل است. فکر نمی کنید بشه بعضی مراحل رو صرفه جویی کرد؟ چطوره که این کار رو بکنیم

```
<?php
function TROut($message) { /*And NO comments about fish, please! ;) */
print "<TR>\n";
$cells=explode("|",$message);
$iterations=count($cells);
$i=0;
while ($i<$iterations) {
list($message,$span)=explode(":",$cells[$i]);
if (strlen($message)<1) $message=" ";
if ($span){
$this->TDOut ($message,$span);
}else{
$this->TDOut ($message);
}
$i++;
}
print "</TR>\n";
}
?>
```

این کار کمی پیچیده تر است. پس بگذارید کمی توضیح بدهم:

خط سوم این عملگر پیغام ها رو بر اساس علامت `Pipe` جدا می کند و هر کدام رو در خانه های آرایه `$cells` قرار می دهد. خط چهارم تعداد پیغام ها را درون `$iterations` قرار می دهد. خط ششم حلقه ما برای حرکت بر روی `cell` ها را شروع می کند. خط هفتم بر `cell` ها را بر اساس علامت دو نقطه ":" تکه می کند و درون متغیرهای `$message` و `$span` قرار می دهد. خط هشتم چک می کند که آیا پیغامی وجود دارد یا نه. در صورتی که نباشد مقدار اولیه را درون آن قرار می دهد. خط نهم نگاه می کند در صورتیکه `span` وجود داشته باشد پیغام را با مقدار `span` مربوطه در خروجی توسط عملگر `TDOut` که قبلا تعریف کردیم قرار می دهد و در صورتیکه `span` وجود نداشته باشد بدون انتقال `$span` همان عملگر را فراخوانی می کند که مقدار اولیه که ۱ است قرارداد شده. در آخر هم که سطر را تمام می کنیم ای عملگر این معنا را می دهد که ما می توانیم فقط یک رشته را به این عملگر بدهیم که تمامی اطلاعات لازم برای نوشتن یک سطر را دارا باشد که دارای تنظیم زیر است

```
"celldata[:colspan]|celldata[:colspan]|.....celldata[:colspan]"
```

پس بجای کاری که قبل از این انجام دادیم می توانیم عناوین و مقادیر یک جدول را اینگونه نشان دهیم:

```
<TABLE>
<?php
```

```
$Theader->TROut ("Name:2 | Address:3" );  
$Theader->TROut ("First|Last|City|State/Province|Country" );  
$Tbody->TROut ("Rod|Kreiser|Cortlandt|New York|USA" );  
?>  
</TABLE>
```

می بینید که کار بسیار ساده تر شد . اما اگر همه اطلاعات درون یک متغیر از نوع آرایه وجود داشت چه ؟
به راحتی می توانیم آنها را درون یک متغیر جدید تلفیق کنیم :

```
<?php  
$message=join($arry, "| " );  
$Tbody->TROut ($message) ;  
?>
```

مطمئناً اطلاعات `span` را نیز نمی توانید تنها با یک تلفیق انجام دهید . باید از تلفیق های بیشتری استفاده کنید . فرض کنید که آرایه شما شش عنصر را در بر می گیرد و فرض کنید سومی و پنجمی باید ۲ و ۳ ستون را به ترتیب `span` کنند . برای این کار می توانید "#:" را در یک آرایه کمکی تلفیق کنید

```
<?php  
$newarray=$arry;  
$newarray[2]=join(list($newarray[2], "2"), ":" );  
$newarray[4]=join(list($newarray[4], "3"), ":" );  
$message=join($newarray, "| " );  
$Tbody->TROut ($message) ;  
?>
```

مشخص است که امکانات بهتری نیز می تواند به این کلاس اضافه گردد .
برگرفته از مقاله ای توسط Rod Kreiser