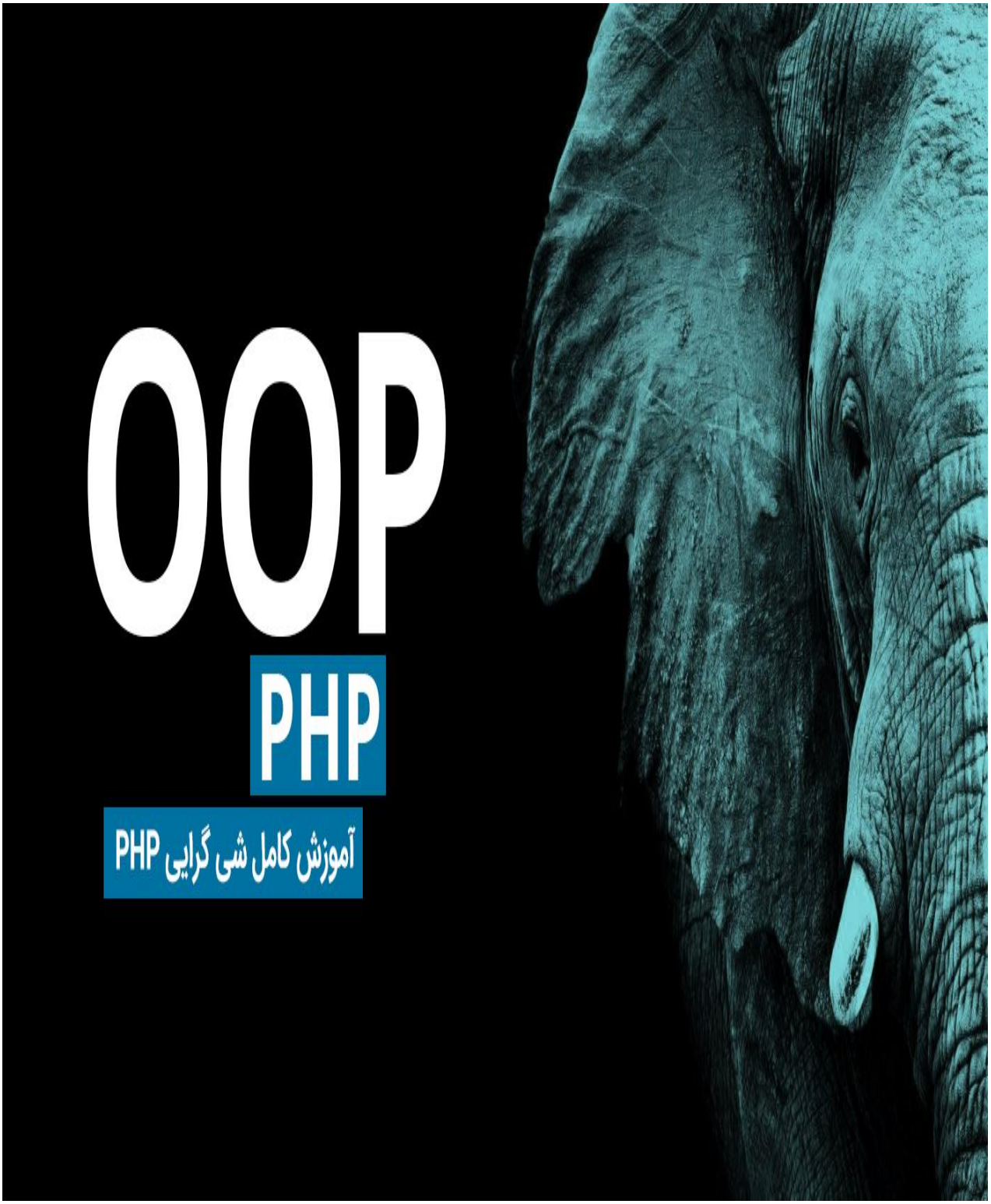

بهار ۹۹

فهرست مطالب

آموزش شی گرایی در PHP – مقدمه

در این قسمت از آموزش های پی اچ پی، به آموزش شی گرایی در PHP خواهیم پرداخت. شما با کلیه مفاهیم OOP و نحوه تعریف و استفاده از کلاس، متد و وراثت در PHP آشنا خواهید شد.

در ادامه با روش استفاده از this در برنامه آشنا می گردید. همین طور روش کار access modifierها و نحوه استفاده از ثابت ها و متدهای آماده را می آموزید.

همچنین با انواع دسترسی به متغیرهای و متدهای یک کلاس و مفاهیم namespace آشنا خواهید شد. ضمن اینکه با روش کار کلاس های abstract و interface و متدهای زنجیره ای آشنا می گردید.

هدف ما در این آموزش، پوشش کلیه مفاهیم شی گرایی و نحوه استفاده از مفاهیم آن در ساختار برنامه می باشد. سعی شده است برای درک کامل از مفاهیم شی گرایی، مثال های کاملی از نحوه استفاده از آن در پی اچ پی، ارائه گردد.

بخش اول: آشنایی با مفاهیم شی گرایی در برنامه نویسی

می توان گفت که برنامه نویسی شی گرا یا همان Object-Oriented Programming یک سبک یا الگو نوشتن کدهاست که به توسعه دهنده ها اجازه می دهد تا بخش های مشابه به هم را در مفهومی به اسم کلاس، گروه بندی نمایند.

در واقع این گروه بندی به برنامه نویسان کمک خواهد کرد که به مفهوم DRY یا همان Don't Repeat Yourself بیشتر نزدیک گردند، مضاف بر اینکه کدهای آن ها بسیار برای نگهداری و تغییر راحت تر خواهد بود. (به این مفهوم easy-to-maintain گفته می شود)

یکی از مهم ترین مزیت های این سبک از کدنویسی، این می باشد که اگر قرار باشد تکه ای از داده های برنامه شما تغییر یابد، عموما لازم است فقط یک بخش در کدهای شما بروز رسانی گردد.

مهم ترین مزیت برنامه نویسی شی گرا که باعث شده است اکثر developer ها از آن استفاده کنند، شبیه سازی برنامه نویسی با مفاهیم دنیای واقعی است. در واقع شما مفاهیم بسیار پیچیده برنامه نویسی را می توانید با مفاهیم شی گرایی به مفاهیم جاری و قابل فهم در دنیا واقعی تبدیل کنید.

همین طور که می دانید، اطراف ما از اشیا مختلف تشکیل یافته است. هر کدام از آن ها خواصی دارند که آن را از شی مشابه خود، متفاوت می کنید. ضمن اینکه ارتباط بین اشیا با فرستادن پیام بین همدیگر برقرار می گردد.

هر شی می تواند رفتاری از خود بروز دهد که مختص همان شی می باشد. این مفاهیم باعث شد تا همه این ها در مفاهیم برنامه نویسی شی گرا استفاده شود.

در ادامه به طور مفصل در این مورد صحبت خواهد شد. در واقع برنامه نویسی با پیروی از الگوهای OOP ، یک روش برنامه نویسی است که در آن برنامه نویس، تمامی متغیرها و توابعی که بهم مرتبط می باشند را در قالب یک کلاس منفرد، سازمان دهی می کند.

مثل تمام الگوهای برنامه نویسی، این الگو نیز مزایا و معایب خود را دارد که در نوشتار نمی گنجد. به صورت کوتاه می توان گفت ممکن است که در پروژه های ساده و همین طور در سایز کوچک، استفاده از الگوی Procedural که بر مبنای استفاده از توابع می باشد، به صرفه تر باشد.

اما در پروژه هایی با ابعاد بزرگ و پیچیده، مسلما باید از الگوهای شی گرا در برنامه نویسی استفاده کرد چرا که هم باعث کاهش پیچیدگی کدها در آینده خواهد شد و هم نگهداری و عیب یابی آن به مراتب ساده تر از روش های برنامه نویسی رویه ای می باشد.

مهم ترین مفاهیم الگو شی گرایی عبارتند از:

- **کلاس Class –**
- **شی Object –**
- **خاصیت ها و رفتار کلاس Property and Method –**
- **ارث بری یا وراثت Inheritance –**
- **چند ریختی Polymorphism –**
- **کپسوله سازی Encapsulation –**

درک مفهوم کلاس و شی

قبل از اینکه وارد مفاهیم عمیق تر در شی گرایی گردیم، ابتدا درک مفاهیم کلاس و اشیا بسیار حیاتی می باشد. دو مفهوم class و object ، مفاهیمی هستند که خیلی اوقات اشتباه توسط توسعه دهندگان به کار برده می شود یک کلاس می تواند یک طرح کلی و اولیه از یک خانه باشد در واقع کلاس شکل و اندازه های بخش های مختلف یک خانه را بیان می کند. اما در واقع این طرح کلی درواقع یک خانه واقعی نیست.

در واقع این پلن اولیه، اصلا وجود خارجی ندارد، اما کاملا مشخص می کند که یک خانه چطور و چگونه باید ساخته شود. در این طرح اولیه به صورت کامل شرح داده می شود که یک خانه چه اجزایی دارد و آن ها چگونه باید در داخل خانه جایگذاری گردند.

در واقع یک شی، یک نمونه واقعی از خانه ساخته شده یا خانه ای است که از روی نقشه خانه ساخته شده است. این خانه دقیقا از روی نقشه تهیه شده و اجزا آن کاملا مانند blueprint اولیه می باشد.

در واقع داده هایی که در این شی یا خانه واقعی ذخیره شده، همان چوب و سنگ و سیمان و سیم ها و سایر اجزایی است که یک خانه از آن ها تشکیل شده است. این اجزا بدون قرار گرفتن در کنار هم و اسمبل نشدن، در واقع مفهمومی به نام خانه را به وجود نخواهند آورد.

تعریف کلاس و اشیا:

کلاس ها ساختارهای داده ها و عملیات مرتبط به آن ها را گروه بندی می کنند و می توان از اطلاعات یک کلاس برای ساختن اشیا استفاده کرد.

اشیا در واقع نمونه های واقعی از یک کلاس هستند که دارای خواص آن کلاس به همراه داده های واقعی و همین طور رفتار مشابه با کلاس هستند.

بخش دوم: تعریف کلاس در PHP و ساخت اشیا

اولین بخش از مفاهیم شی گرایی در PHP شامل نحوه تعریف کلاس، اشیا و نحوه دسترسی به آن ها می باشد. در این قسمت ابتدا به تعریف کلاس در پی اچ پی اشاره خواهیم کرد. سپس می آموزیم که چگونه یک شی از آن ساخته می شود.

پس از آن به تعریف متد و خاصیت های یک کلاس و سطح دسترسی به آن ها خواهیم پرداخت. در ادامه به تشریح متدهای جادویی و بخش مهم آن یعنی متد construct خواهیم پرداخت.

تعریف کلاس و ساخت شی از آن در PHP

برای تعریف یک کلاس باید از کلمه کلیدی class به صورت روبرو استفاده کنیم. برای ساخت شی از آن کافی است کلمه کلیدی new استفاده کنیم:

```
1. class MyClass
2. {
3. // Class properties and methods
4. }
5. $obj = new MyClass;
```

برای اینکه با ساختار تعریف متد و همین طور خواص در یک کلاس آشنا شوید، از یک مثال استفاده کرده ایم. در ادامه توضیحات مربوط به این مثال آمده است.

در صورتی که کدهای زیر را در یک فایل با پسوند php ذخیره کنید، می توانید به راحتی آن را در localhost خود و با استفاده از WAMP یا XAMP یا MAMP به اجرا در آورید:

```
1. //create a class
2. class MsnCar {
3.
4. //define properties in a class
5. public $name;
6. private $wheel_count = 4;
7. private $door_count;
8. protected $company;
9. public static $car_static_property = "This is for parent class";
```

```
10.
11. /**
12. * MsnCar constructor.
13. * @param $name
14. * @param int $wheel_count
15. */
16. public function __construct($name=null, $wheel_count=null)
17. {
18. $this->name = $name;
19. $this->wheel_count = $wheel_count;
20. }
21.
22.
23. //Sample of setter and getters
24. /**
25. * @return mixed
26. */
27. public function getCompany()
28. {
29. return $this->company;
30. }
31.
32. /**
33. * @param mixed $company
34. */
35. public function setCompany($company)
36. {
37. $this->company = $company;
38. }
39.
40.
41. /**
42. * @return int
43. */
44. public function getWheelCount()
45. {
46. //Sample of property referencing in a class
47. return $this->wheel_count;
48. }
49.
50. /**
51. * @param int $wheel_count
52. */
53. public function setWheelCount($wheel_count)
54. {
55. $this->wheel_count = $wheel_count;
```

```
56. }
57.
58. /**
59. * @return mixed
60. */
61. public function getDoorCount()
62. {
63. return $this->door_count;
64. }
65.
66. /**
67. * @param mixed $door_count
68. */
69.
70. public function setDoorCount($door_count)
71. {
72. $this->door_count = $door_count;
73. }
74.
75.
76. //create a method in a class
77. function moving() {
78. echo "Now I'm moving So fast".'<br>';
79. }
80.
81. function stopping() {
82. echo "Now I'm stopping so well".'<br>';
83. }
84.
85. //using properties object inside a class method
86. function car_details() {
87. echo $this->name.' has '.$this->wheel_count.' wheels and '.$this->door_count.' doors';
88. }
89.
90. //Sample of static method to referencing parent class
91. public static function car_reference()
92. {
93. echo self::$car_static_property.'<br>';
94. //sample of late static binding
95. echo static::$car_static_property.'<br>';
96.
97. }
98.
99. }
100.
101.
```

```
102.    //Instantiating of a class
103.    $bmw = new MsnCar();
104.    $benz = new MsnCar();
105.    $myPride = new Pride();
106.
107.    //Invoke property of an object
108.    echo $bmw->getWheelCount()."<br>";
109.
110.    //change or assign new value to object properties
111.    $bmw->setDoorCount(2);
112.    $benz->setWheelCount(6);
113.    echo $bmw->getDoorCount()."<br>";
114.    echo $benz->getWheelCount()."<br>";
```

تعریف خصوصیات کلاس در PHP

Propertyیا خاصیت برای تعریف متغیر یا اضافه کردن داده به یک کلاس مورد استفاده قرار می گیرد. همان طور که می بینید برای تعریف Property در یک کلاس از همان تعریف متغیر در پی اچ پی استفاده می شود، تنها تفاوت آن این می باشد که هر متغیر دارای یک سطح دسترسی است که جلوتر در مورد آن توضیح داده خواهد شد.

توجه داشته باشید که برای تعریف یک ثابت، می توانید از کلمه کلیدی const استفاده کنید. همان طور که می بینید، کلاسی با نام MsnClass تعریف شده است که دارای Property ها (یا خاصیت هایی) به شکل زیر می باشد:

```
1.  //define properties in a class
2.  public $name;
3.  private $wheel_count = 4;
4.  private $door_count;
5.  protected $company;
6.  public static $car_static_property = "This is for parent class";
7.  const site="miladsamani.ir";
```

تعریف متد های کلاس در PHP

Methodها در واقع توابعی هستند که در یک کلاس تعریف شده و مورد استفاده قرار می گیرند. همان طور که می بینید برای تعریف متد در یک کلاس از همان تعریف تابع در PHPاستفاده می شود، تنها تفاوت آن این می باشد که هر متد دارای یک سطح دسترسی است که جلوتر در مورد آن توضیح داده خواهد شد.

همان طور که می بینید، کلاسی با نام MsnClass تعریف شده است که دارای متدهایی به شکل زیر می باشد:

```
1. //create a method in a class
2. public function moving() {
3. echo "Now I'm moving So fast".'<br>';
4. }
5.
6. public function stopping() {
7. echo "Now I'm stopping so well".'<br>';
8. }
```

روش استفاده از متغیرها و متدهای یک کلاس در PHP

برای استفاده از متغیرها و متدهای یک کلاس حتما باید یک نمونه یا شی از آن کلاس ساخته شود تا بتوان متغیر های آن کلاس را مقداردهی کرده و از متدهای آن استفاده کرد. (این مورد در مورد متغییر و متد استاتیک صادق نیست)

نحوه مقدار دهی به متغیرها، همانند مقداردهی به متغیر در php می باشد، با این تفاوت که باید حتما از نام شی و سپس علامت class accessor و سپس نام متغیر استفاده نمود. برای توابع یا متد هم به همین صورت می باشد و اینکه حتما باید از () و در صورت نیاز، تعریف پارامترهای ارسالی به تابع، استفاده کرد.

```
1. //Instantiating of a class
2. $bmw = new MsnCar();
3. $benz = new MsnCar();
4. $myPride = new Pride();
5.
6. //Invoke property of an object
7. echo $bmw->getWheelCount()."<br>";
8. echo $myPride->moving();
```

متغیر استاتیک و متد استاتیک در یک کلاس:

در صورتی که یک متغیر یا یک کلاس در بین همه اعضای یک کلاس مشترک باشد، آن را از نوع استاتیک تعریف می کنند. برای استفاده از یک متغیر استاتیک لازم نیست که حتما شی ای از کلاس ساخته شود.

متغیر استاتیک و همین طور متدهای استاتیک با استفاده از نام کلاس و استفاده از علامت Scope Resolution یا همان :: فراخوانی می شوند

نکته مهم در مورد متغیرها و متدهای استاتیک

برای استفاده از متغیرها و متدهای یک کلاس، حتما می باید یک شی از آن ساخته شود اما برای استفاده از متغیر یا متد استاتیک، نیازی به ساخت یک شی نیست. در واقع برای بار اول که کلاس در حافظه ایجاد می گردد یا instantiate می گردد، شما به راحتی می توانید از نام کلاس به همراه علامت scope resolution برای استفاده از آن بهره ببرید.

از آنجایی که ساخت شی در حافظه مستلزم صرف حافظه می باشد، لذا استفاده از متغیرها و متدهای استاتیک، باعث کاهش چشمگیر حافظه مصرفی برنامه شما خواهد شد. عموما متغیرهای استاتیک ساخته شده به عنوان counter یا شمارنده مورد استفاده قرار می گیرند. همچنین متدهای استاتیک، عموما در کلاس های utilityیا کمکی که جهت سرویس دهی به کلاس اصلی می باشند، ساخته می شوند.

کلاس های کمکی عموما کارهایی مثل تبدیل واحد، encryptionیا رمزنگاری و sanitationرا به انجام می رسانند.

```
1.  //create a class
2.  class MsnCar {
3.  //define static property in a class
4.  public static $car_static_property = "This is a static property in a class";
5.
6.
7.  //Sample of static method in a class
8.  public static function moving()
9.  {
10. echo 'Hey, All of cars move!<br>';
11. }
12. }
13.
14. echo MsnCar::car_static_property;
15. MsnCar::moving();
```

تعریف سطح دسترسی به متدها و متغیرهای کلاس در PHP

شما می توانند برای متغیر ها و متدهای کلاس خود، تعریف کنید که چگونه قابل دسترس باشند. در واقع شما می توانید قابلیت دیده شدن آن ها در خارج و داخل کلاس را تعریف نمایید.(Assigning Visibility)

عموما در مفاهیم شی گرایی، خواص یا متغیر های یک کلاس به صورت خصوصی تعریف می شوند تا از دستکاری کردن از آن ها در خارج از کلاس جلوگیری شود .از طرف دیگر در ارث بری، ممکن است بخواهیم به خواص یک کلاس والد یا parent و یا بالعکس، دسترسی داشتیم و امکان تغییر در خارج از آن وجود نداشته باشد.

همچنین اکثر متدها برای دسترسی به رفتار یک شی، به صورت عمومی تعریف می شوند. البته این بسته به نوع طراحی کلاس ممکن است متفاوت باشد. بر مبنای همین موارد، سه نوع دسترسی به متغیر ها و متدهای یک کلاس در پی اچ پی، تعریف شده است.

شما با استفاده از کلمات کلیدی زیر، می توانید نحوه دسترسی به یک کلاس را کنترل نمایید:

- **سطح دسترسی به صورت public**

در این نوع سطح دسترسی، متد یا متغیری که با public مشخص می شود، قابل دسترس برای همه (چه از داخل کلاس و چه از خارج کلاس) خواهد بود.

- **سطح دسترسی به صورت private**

در این حالت فقط می توان به متغیر یا متد از داخل کلاس دسترسی داشت .دسترسی از خارج کلاس به متد یا متغیر از نوع private امکان پذیر نیست.

- **سطح دسترسی به صورت protected**

در حالت protected کلاس و همین طور سایر کلاس هایی که از کلاس جاری ارث برده اند (یا همان کلاس های فرزند) می توانند به متغیر یا متدی با این نوع دسترسی داشته باشند.

چرا باید از Access Modifier یا سطوح دسترسی در شی گرایی در PHP استفاده کنیم؟

در واقع با این کار، کلیه اعمال ویرایشی که از خارج کلاس بخواهد بر روی اعضای داخل کلاس (متدها و خاصیت ها) اعمال گردد را، محدود می کنیم. در واقع فقط متدهای تعریف شده یک کلاس اجازه دسترسی و تغییرات بر روی اعضای یک کلاس را دارند.

در واقع از بیرون یک کلاس نمی توان به آن تزریق کرد و ما با محدود سازی و اعتبارسنجی، جلوی این کار را خواهیم گرفت. ما توسط دادن سطح دسترسی، تمامی داده هایی که خارج از کد کلاس، وارد سیستم می شوند را مورد اعتبارسنجی قرار داده و فیلتر می کنیم. بنابراین احتمال آسیب دیدگی برنامه خود را تقریبا به صفر نزدیک و امنیت داده های سیستم را تضمین می کنیم.

```
class MsnCar {
//define properties in a class with several visibility
public $name;
private $wheel_count = 4;
private $door_count;
protected $company;
public static $car_static_property = "This is for parent class";

//Define access modifier for functions
```

```
10. public static function moving() {
11. echo "Now I'm moving So fast".'<br>';
12. }
13. }
```

همانگونه که بیان شد، برای جلوگیری از دسترسی مستقیم به داده های یک کلاس یا چک کردن آن ها مثلا پیش از ذخیره در دیتابیس، عموما متغیرهای یک کلاس را از نوع privateتعریف می کنند.

به همین دلیل، از آنجایی که نمی توان به صورت مستقیم به این متغیر ها دسترسی داشت، باید تابع هایی به عنوان Getter و Setter تعریف نمود تا بتوان متغیرها را از داخل کلاس فراخوانی نمود و یا مقدار آن ها را تغییر داد.

برای دسترسی در داخل کلاس به متغیر ها و متدهای شی ساخته از آن کلاس از کلمه کلیدی با نام this و همین طور علامت ->استفاده می شود. مفاهیم گفته شده به صورت خلاصه در زیر تشریح شده است:

- **استفاده از کلمه کلیدی this برای دسترسی به متدها به متغیرهای یک شی در داخل کلاس:**

برای دسترسی به متغیرهای داخل یک کلاس توسط متدها یا سایر متغیرهای یک شی، از کلمه کلیدی this استفاده می شود.

- **استفاده از کلمه کلیدی self برای دسترسی به متدها به متغیرهای استاتیک در داخل کلاس:**

از آن جایی که متغیر استاتیک متعلق به یک object می باشد، لذا در متدهای استاتیک یا سایر متدها، برای دسترسی به آن نمی توان از کلمه کلیدی this استفاده کرد.

در این موارد برای دسترسی به متدها یا خواص استاتیک در متدهای معمولی، از کلمه کلیدی self به همراه علامت Scope Resolution استفاده می کنیم.

- **استفاده از تابع getter برای دسترسی به متغیر از نوع private در کلاس:**

برای دسترسی به محتوا یا داده داخل یک متغیر از نوع private از توابع getterاستفاده می شود. این تابع ها دارای سطح دسترسی public بوده و با ساخته شدن یک شی، از داخل کلاس به محتوای private دسترسی پیدا می کنند .برای خوانایی بهتر کدهای برنامه، این توابع عموما با نام get آغاز می شوند.

- **استفاده از تابع setter برای تغییر مقدار متغیر از نوع private در کلاس:**

برای تغییر دادن محتوا یا داده داخل یک متغیر از نوع private از توابع setterاستفاده می شود. این تابع ها دارای سطح دسترسی public بوده و با ساخته شدن یک شی، از داخل کلاس به محتوای داده private دسترسی پیدا می کنند. برای خوانایی بهتر کدهای برنامه، این توابع عموما با نام set آغاز می شوند.

```
1.  //Sample of setter and getters
2.  /**
3.  * @return mixed
4.  public function getCompany()
5.  {
6.  return $this->company;
7.  }
8.
9.  /**
10. * @param mixed $company
11. */
12. public function setCompany($company)
13. {
14. $this->company = $company;
15. }
16.
17. /**
18. * @return int
19. */
20. public function getWheelCount()
```

```
21. {
22. //Sample of property referencing in a class
23. return $this->wheel_count;
24. }
25.
26. /**
27. * @param int $wheel_count
28. */
29. public function setWheelCount($wheel_count)
30. {
31. $this->wheel_count = $wheel_count;
32. }
33.
34. /**
35. * @return mixed
36. */
37. public function getDoorCount()
38. {
39. return $this->door_count;
40. }
41.
42. /**
43. * @param mixed $door_count
44. */
45. public function setDoorCount($door_count)
46. {
47. $this->door_count = $door_count;
48. }
49.
50. //using properties object inside a class method
51. function car_details() {
52. echo $this->name.' has '.$this->wheel_count.' wheels and '.$this->door_count.' doors';
53. }
```

متدهای زنجیره ای و روش استفاده از آن در شی گرایی در PHP

گاهی اوقات ممکن است نیاز داشته باشیم که کلاس را خلاصه تر پیاده سازی کرده و همین طور سرعت مقداردهی در آن را بالاتر ببریم.

در اینگونه موارد، برای مقداردهی یکproperty ، از چندین متد در داخل کلاس استفاده کرده و آن ها را به صورت زنجیره ای فراخوانی می کنیم.

در همه توابع یک return داریم در انتها و در متد آخر، مقدار برگشتی را echo می کنیم.

```
1. class Person{
2. protected $person_info;
3. public function name($value){
4. $this->person_info .= "Name: $value, ";
5. return $this;
6. }
7. public function family($value){
8. $this->person_info .= "Family: $value, ";
9. return $this;
10. }
11. public function birthday($value){
12. echo $this->person_info .= "Born on: $value";
13. }
14. }
15. $gholam = new Person();
16. $gholam->name("gholam")->family("Palang")->birthday("1-1-1");
17. /*
18. * The result is:
19. * Name: gholam, Family: Palang, Born on: 1-1-1
20. * */
```

متدهای جادویی و روش استفاده از آن در شی گرایی درPHP

جهت ساده سازی کار برنامه نویسان برای ساختن و استفاده کردن از object های ساخته شده از class ها، چندین متد با نام های متدهای جادویی در زبان PHPقرار داده شده است.

این متدهای با علامت ___ دو underline به صورت پشت سرهمشروع می شوند .معمولا قبل از متدهای جادویی از access modifier های مثل public یا private یا protectedاستفاده نمی شود.

متدهای جادویی در که در شی گرایی در php استفاده می شوند، به شرح زیر می باشند:

1. __construct
2. __destruct
3. __call
4. __callStatic
5. __get
6. __set
7. __isset

و

1. __unset
2. __sleep
3. __wakeup
4. __toString
5. __invoke
6. __set_state
7. __clone

نحوه استفاده از متد جادویی construct در PHP

زمانی که تصمیم داشته باشیم برای ساخت یک شی از یک کلاس، عملیات خاص یا مقداردهی خاصی را انجام دهیم، از این متد استفاده می شود. به این متد اصطلاح سازنده کلاس یا constructor نیز گفته می شود.

عموما برنامه نویسان برای مقدار دهی اولیه به متغیرهای شی ساخته شده)برای جلوگیری از null بودن مقادیر آن ها (از این متد استفاده می کنند .توجه داشته باشید که متد

همین که یک object جدید از یک کلاس ساخته می شود، به صورت اتوماتیک فراخوانی می شود.

در داخل متد construct نمی توان از return نیز استفاده کنید. در واقع هرگاه که بخواهیم دقیقا پس از ساخته شدن شی، کاری را روی آن انجام دهیم، از متد جادویی constructاستفاده می کنیم.

```
1.  //create a class
2.  class MsnCar {
3.  //define properties in a class
4.  public $name;
5.  private $wheel_count = 4;
6.  private $door_count;
7.  protected $company;
8.  public static $car_static_property = "This is for parent class";
9.  /**
10. * MsnCar constructor.
11. * @param $name
12. * @param int $wheel_count
13. */
14. public function __construct($name="pride", $wheel_count=4)
15. {
16. $this->name = $name;
17. $this->wheel_count = $wheel_count;
18. }
19. }
20.
21. $benz = new MsnCar("benz1",6);
```

- **نحوه استفاده از متد جادویی destruct در PHP**

این مفهوم دقیقا عکس مفهوم construct عمل می کند، در واقع متد destruct دقیقا قبل از اینکه شی ساخته شده از یک کلاس، از بین برود، فراخوانی می شود. این متد با عنوان Destructor یا از بین برنده برای object ساخته شده از یک کلاس، شناخته می شود.

این متد دقیقا پیش از زمانی که شی در حافظه destroy می شود (یا از بین می رود) به اجرا در می آید. به عنوان مثال زمانی که شما متد unset را بر روی یک شی فراخوانی می کند، این متد پیش از unset کردن شی مورد نظر، اجرا می گردد.

در نظر داشته باشید متد جادویی destruct نمی تواند پارامتر ورودی دریافت نماید.

```
1.  class MsnCar
2.  {
3.  function __construct()
4.  {
5.  echo 'It is called after creating an object from MsnCar class';
6.  }
7.
8.  function __destruct()
9.  {
10. echo 'It is called before destroying an object from MsnCar class';
11. }
12. }
```

نحوه استفاده از متد جادویی tostring در PHP:

در صورتی که بر روی شی ساخته شده از یک کلاس، تابع echo را فراخوانی کنیم، این متد در کلاس به اجرا در می آید.

نحوه استفاده از متد جادویی set در PHP:

این متد بر روی یک object زمانی فراخوانی می شود، که بخواهیم مقداری را به یک propertyبر روی شی نسبت دهیم که این خاصیت یا در کلاس مربوط به آن شی وجود ندارد یا دسترسی لازم برای تغییر آن برای ما وجود ندارد.

در واقع از این متد می توان برای ساختن متغیر داینامیک برای یک کلاس بهره برد. متد جادویی set دو آرگومان ورودی می پذیرد که اولی نام خاصیت مورد نظر و دومی مقداری می باشد که قرار است به آن اختصاص دهیم.

نحوه استفاده از متد جادویی get در PHP:

برای دسترسی به متغیرهایی که با روش بالا و استفاده از متد جادویی set تعریف شده اند، می توانید از متد جادویی get استفاده نمایید. در واقع این متد زمانی فراخوانی می شود که ما بخواهیم به یک property دسترسی باشیم که یا در کلاس ما وجود ندارد و یا سطح دسترسی به آن وجود ندارد.

به عنوان مثال ما می توانیم از بیرون کلاس با متد جادویی get خاصیت هایی از کلاس که به صورت private تعریف شده اند را نمایش بدهیم.

نحوه استفاده از متد جادویی isset در PHP:

برای چک کردن اینکه property های غیر قابل دسترس در یک کلاس یا همان private، یا property هایی که به صورت داینامیک در کلاس تعریف شده اند، مقداردهی اولیه شده اند، از متد جادویی isset استفاده می کنیم.

نحوه استفاده از متد جادویی unset در PHP:

برای unset کردن property هایی که در قسمت های قبل صحبت کردیم از متد جادویی unset در php استفاده می کنیم.

نحوه استفاده از متد جادویی clone در PHP:

زمانی که بخواهیم از یک شی با استفاده از روش copy by value یک کپی تهیه کنیم، از متد جادویی clone استفاده می کنیم.

بخش سوم: ارث بری در PHP

در این بخش می خواهیم به مفاهیم وراثت در پی اچ پی بپردازیم. در ابتدا نیاز داریم که با مفهوم ارث بری و اینکه چرا در شی گرایی، به ارث بری نیاز داریم بپردازیم.

می توان گفت از مهم ترین مزیت هایی که باعث می شود به سمت برنامه نویسی شی گرا سوق پیدا کنیم، کاهش نوشتن خط های اضافی کد و کم شدن کدهای تکراری در برنامه ما می باشد. در واقع ما با استفاده از وراثت به مقدار بسیار زیادی از code duplication جلوگیری خواهیم کرد.

در مفهوم Inheritance از یک کلاس والد یا همان parent class استفاده می شود .این کلاس دارای تعدادی خواص و رفتارهای مربوط به خود می باشد. حال کلاسی به نام کلاس فرزند یا child classتعریف می گردد که می تواند تمامی خواص و رفتارهای کلاس والد خود را به ارث ببرد.

علاوه بر این، کلاس فرزند می تواند property ها و method های جدیدی برای خودش داشته باشد که در کلاس والد وجود ندارد. همچنین کلاس فرزند می تواند رفتار و خواص متد والد خود را مورد بازنویسی قرار دهد

در واقع مفهوم وراثت این امکان را به ما می دهد که کدهای خود را تنها یک بار در کلاس پدر نوشته و به کرات آن را در کلاس های فرزندی که از کلاس پدر ارث برده اند، استفاده نماییم.

در مورد وراثت و نحوه ارث بری در PHP نکاتی وجود دارد که در زیر به آن ها اشاره شده است:

- **نحوه ارث بری یک کلاس از کلاس دیگر در PHP**

برای اینکه بتوانید از یک کلاس ارث ببرید، کافی است از کلمه extends به صورت زیر استفاده نمایید. در نظر داشته باشید، با ارث بردن از کلاس والد، کلیه خاصیت ها و متدهای کلاس parentبرای کلاس child هم به همان شکل تعریف می گردند.

```
1.  class MsnCar
2.  {
3.  //define properties in a class
4.  public $name;
5.  protected $company;
6.  private $wheel_count = 4;
7.  private $door_count;
8.  public function __construct($name, $door_count, $company)
9.  {
10. $this->name = $name;
11. $this->door_count = $door_count;
12. $this->company = $company;
13. }
14. public function moving()
15. {
16. echo "Now I'm moving So fast" . '<br>';
17. }
18. public function stopping()
```

```
{
echo "Now I'm stopping so well" . '<br>';
}
public function __toString()
{
return 'The name of car is ' . $this->name . ' and from ' . $this->company . ' with ' . $this->door_count . ' doors!!!<br>';
}
}

//Inheritance in PHP
class PrideGhahreman extends MsnCar
{
public $wings_number;
public function __construct($name, $door_count, $company)
{
parent::__construct($name, $door_count, $company); // Call the parent class's constructor
echo "A new constructor in " . __CLASS__ . ".<br />";
}
public function parvaz()
{
echo "You can fly with " . __CLASS__ . ".<br />";
}
public function __toString()
{
$temp = parent::__toString();
return $temp.' This Pride has '.$this->wings_number.' wings';
}
}

// Create a new object
$newobj = new PrideGhahreman('pride 6 dar', 6, 'Saipa Ghashang');

// Define and use new method in child class
echo $newobj->parvaz();

// Use a method from the parent class
$newobj->moving();

// Use a method from the parent classg
$newobj->stopping();

// Define and use new property in child class
$newobj->wings_number = 2;

```

```
63. // Use a method from the parent class and ovverriding parent method in child method
64. echo $newobj;
65.
66.
67. /* The result is:
68. A new constructor in PrideGhahreman.
69. You can fly with PrideGhahreman.
70. Now I'm moving So fast
71. Now I'm stopping so well
72. The name of car is pride 6 dar and from Saipa Ghashang with 6 doors!!!
73. This Pride has 2 wings
```

سطوح دسترسی در استفاده از وراثت در PHP

همانند آنچه در قبل تر در مورد سطح دسترسی ها در PHP بیان شد، کلاس فرزند می تواند تنها به متد ها و property هایی با سطح دسترسی public و protectedدر کلاس پدر خود دسترسی داشته باشد. کلاس child دسترسی به متدها یا خاصیت های private در کلاس parentرا نخواهد داشت.

در نظر داشته در صورتی که متد یا خاصیتی به صورت protected تعریف گردد، امکان دسترسی به آن هم در داخل کلاس والد و هم در کلاس های ارث برده شده از آن، فراهم می باشد.

تعریف property ها و method های اختصاصی در کلاس child

یک کلاس child یا فرزند، علاوه بر اینکه به متدها و خواص کلاس parent خود دسترسی دارد، می تواند دارای متدها و همین طور خاصیت های مربوط به خود باشد. در مثال بالا برای کلاس فرزند، این امر نمایش داده شده است.

مفهوم overriding در PHP

در صورتی که برای متد (یا خواص) موجود در کلاس والد، پیاده سازی جدیدی در کلاس فرزند آن کلاس داشته باشیم، در این صورت می گوییم آن متد یا خاصیت overrideشده است. در صورتی که متد یا خاصیتی override شده باشد، برای دسترسی به متدهای والد، باید از کلمه کلیدی parent استفاده کنید.

همانگونه که در مثال بالا می بینید، متد construct در کلاس child دوباره نوشته شده و متد constructکلاس parent در داخل آن با کلمه کلیدی parent و استفاده از علامت scope resolutionیا :: فراخوانی شده است.

مفهوم :overloading تفاوت overloading با overriding چیست؟

در بسیاری از مواقع این دو مفهوم باهم اشتباه گرفته شده و اسامی آن ها به جای همدیگر به کار برده می شود. در مفهوم overriding که در بالا در مورد آن توضیح داده شد، دو متد دقیقا با یک نام و یک تعداد آرگومان ورودی اما با پیاده سازی متفاوت، در کلاس پدر و در کلاس فرزند موجود می باشند.

اما درoverloading ، چند متد در یک کلاس با یک نام اما با تعداد آرگومان های ورودی متفاوت، تعریف و پیاده سازی می شوند (این امر بسیار در مورد constructorها استفاده می شود

نحوه استفاده از کلمه کلیدی Final برای متد ها درPHP

در صورتی که بخواهید از override شدن یک متد در کلاس فرزند، جلوگیری به عمل آورید، کافی است در ابتدای تعریف متد در کلاس والد، از کلمه کلیدی Finalاستفاده کنید.

نحوه استفاده از کلمه کلیدی Final برای کلاس درPHP

در صورتی که بخواهید اجازه ارث بری از یک کلاس را بگیرید و نگذارید تا آن کلاس extend گردد، از کلمه کلیدی final در پیش از کلمه class در تعریف کلاس، استفاده می کنید.

شما نمی تواند از خاصیت inheritance بر روی کلاس های final استفاده کنید و نمی توانید از آن ها ارث بری کنید.

مفهوم late static binding درPHP

مفهوم late static binding از مفاهیم بسیار مهم و کاربردی در php می باشد. در پیش تر گفته شد که برای دسترسی به یک متد یا خاصیت از نوع استاتیک در متدهای یا خاصیت معمولی، می باید به جای استفاده از کلمه کلیدی this از کلمه کلیدی self استفاده کنیم.

حال گاهی اوقات ممکن است در کلاس parent بخواهیم از متد یا property ای که به صورت staticدر کلاس فرزند تعریف شده، استفاده نماییم. به این کار اصطلاحا late static binding گفته می شود.

برای این کار، یعنی استفاده از متد یا خاصیت static کلاس فرزند در کلاس پدر، ار کلمه کلیدی staticبه جای self استفاده می کنیم. در مقاله ای جداگانه و به صورت کاملا کاربردی، این مورد با مثال به شما نمایش داده خواهد شد (در هنگام ساخت یک کلاس برای دسترسی به دیتابیس و کوئری زدن بر روی آن)

آشنایی با مفهوم auto loading در PHP

مفهوم autoloading یکی از موارد بسیار مهم و کاربردی در PHP می باشد .همواره برای اینکه بتوانیم از یک فایل یا یک کلاس PHP در فایل ها یا کلاس های دیگر استفاده نماییم، می باید محل آن فایل را با دستور include یا requireبه بالای فایل خود اضافه کنیم.

این امر وقتی برنامه کوچک است و تعداد بخش ها و فایل های آن زیاد نیست، مهم نمی باشد. اما هنگامی که برنامه کمی بزرگ می شود و تعداد بخش های آن افزایش می یابد، این امر بسیار مشکل شده و هر بار می باید تعداد زیادی فایل را قبل از استفاده، includeنماییم.

زمانی که تعداد فایل ها افزایش می یابد، مدیریت این کار فوق العاده سخت و پیچیده خواهد شد. در واقع ما باید تمام فایل ها لود کنیم در حالی که فقط به بخشی از آن ها نیاز داریم. در ابتدا برای مدیریت این کار از متد جادویی autoloadبرای کلاس ها استفاده می شد.

در PHP7 و در SPL یا همان Standard PHP Library تابعی با نام spl_autoload_register معرفی شد. در واقع PHP در صورتی که فایل یا کلاسی را در برنامه ببیند که در ابتدای برنامه تعریف نشده، آن را به این تابع ارجاع می دهد تا این تابع به صورت خودکار، فایل یا کلاس مورد نظر را پیدا کرده و به برنامه اضافه کند.

به این عمل اصطلاحا autoloading گفته می شود. مزیت استفاده از این روش این می باشد که هم استفاده از فضای حافظه کمتر خواهد شد و هم باعث افزایش سرعت پردازش برنامه می گردد.

برای انجام auto loading و استفاده از تابع spl_autoload_register ، شما می باید یک ساختار مشخص برای فایل ها یا کلاس های خود مشخص کنید، تا هنگام ارجاع شدن یک فایل یا کلاس ناشناخته، کلاس autoloading شما بتواند ساختار آن را درست کرده و به برنامه شما اضافه کند.

به عنوان مثال، شما در کلیه کلاس های خود را در یک دایرکتوری می گذارید .همچنین اول اسم کلیه کلاس ها را با class- نام گذاری می کنید.

حال فقط لازم است کلاس autoloader را خود را در هدر صفحات یا تابعی به نام init.php اضافه کنید. پس از آن دیگر نیاز به اضافه کردن فایل ها و کلاس های خود برای شناخته به برنامه ندارید و کلاس autoloader شما به راحتی به صورت خودکار این کار را برای شما انجام خواهد داد.

کد زیر، یک نمونه از کلاس autoloader می باشد. البته در نظر بگیرید که پس از تعریف این کلاس، کلیه کلاس های شما باید با نام class- آغاز گردد تا کلاس autoloader شما بتواند آن ها را به صورت اتوماتیک به برنامه در حال اجرای شما اضافه نماید.

البته روش های دیگری مثل استفاده از composer با استفاده از PSR-0 و PSR-4وجود دارد که در این مقاله نمی گنجد. حتما در بخش بعدی این مقاله یک مثال عملی و واقعی از این مورد، ارائه خواهد شد.

```
1.  /***** Autoloder class *****/
2.  class Autoloader
3.  {
4.  public function __construct()
5.  {
6.  spl_autoload_register(array($this, 'autoload'));
7.  }
8.  public function autoload($class_name)
9.  {
10. $file = $this->convert_class_to_file($class_name);
11. if (is_file($file) && file_exists($file) && is_readable($file) &&
    !class_exists($class_name)) {
12. //var_dump($file);
13. include $file;
14. }else {
15. die("This file name: {$file} was not found...!");
16. }
17. }
18. public function convert_class_to_file($class_name)
19. {
20. $class = strtolower($class_name);
21. $class = 'class-'.$class;
22. $filename = "includes/{$class}.php";
23. return $filename;
24. }
```

```
25. }
26. new Autoloader();
```

بخش چهارم: مفهوم Abstract شی گرایی در PHP

در مفاهیم شی گرایی، زمانی که بخواهیم سایر توسعه دهندگان را الزام کنیم که در هنگام ارث بری از کلاسی، حتما متد خاصی در آن را پیاده سازی کنند، از کلاس ها و متدهای abstract استفاده می کنیم.

در واقع در یک کلاس abstract یا انتزاعی می خواهیم که برنامه نویس حتما متدی که در کلاس والد با نام abstract مشخص شده است را، پیاده سازی نماید.

اگر یک متد abstract در یک کلاس موجود باشد، حتما آن کلاس باید به صورت abstract تعریف گردد. کلاس abstract قابل نمونه گیری یا instantiation نبوده اما قابل ارث بری می باشند.

متدهای انتزاعی در داخل کلاسabstract ، دارای فقط اسم و آرگون های ورودی هستند و بدنه آن ها در کلاس abstract یا والد خالی است. در واقع کلاس انتزاعی والد، فقط دارای Method's Signatureمی باشد.

بنابراین بدنه متدهای انتزاعی و عملیاتی که قرار است در آن ها انجام شود، در کلاس فرزند که از آن ارث برده است، می باید تعریف گردد. در واقع implementationمتد در این بخش باید انجام پذیرد.

در نظر داشته باشید که Visibility یا همان سطح دسترسی متد فرزند، باید مساوی یا بیشتر از سطح دستری متد abstract باشد. یعنی اگر visibility متد انتزاعی در کلاس parent برابر protectedباشد، در کلاسی که از آن مشتق شده است، سطح دسترسی نمی تواند private بوده و باید protected یا public باشید.

کلاس های abstract می توانند دارای property ها یا متد هایی باشند که پیاده سازی شده باشند. یعنی یک کلاس انتزاعی می تواند شامل خاصیت ها و همین طور متدهای غیر انتزاعی نیز باشد.

نحوه پیاده سازی کلاس abstract و متد abstract در PHP:

برای اینکه یک کلاس انتزاعی تعریف کنید، کافی است یک یا چند متد abstractدر یک کلاس که با کلیدی واژه abstract آغاز شده است را، به صورت زیر تعریف نمایید.

- **تعریف property و method های غیر انتزاعی در PHP:**

همانند یک کلاس معمولی، شما می توانید در داخل یک کلاس abstract ، خواص و متدهای غیر abstract تعریف نمایید.

```
1. abstract class AbstractMsnCar
2. {
3. /*Abstract classes can have property*/
4. protected $tankCapacity;
5.
6. /*Abstract classes can have non abstract method*/
7. public function setTankCapacity($tankCapacity)
8. {
9. $this->tankCapacity = $tankCapacity;
10. }
11.
12. // Abstract method
13. abstract public function distanceOnFullTank();
14. }
```

نحوه ارث بری یک کلاس از کلاس abstract در PHP

ارث بری از یک کلاس انتزاعی، همانند ارث بری در سایر بخش هاست. فقط حتما باید متدهایی که در کلاس parent تعریف شده اند، در کلاس child بازنویسی شده یا implement گردند.

متد abstract در کلاس والد، هیچ بدنه ای ندارد و در کلاس فرزند، این متد باید override گردد.

```
1. class VanetPride extends AbstractMsnCar
2. {
3. public function distanceOnFullTank()
4. {
5. // TODO: Implement distanceOnFullTank() method.
6. $kilometers = $this->tankCapacity*100;
7. return $kilometers;
8. }
9.
10. }
```

مفهوم Encapsulation شی گرایی و استفاده از namespace در PHP

یکی دیگر از مفاهیم شی گرایی در PHP ، مفهوم Encapsulation و نحوه پیاده سازی آن می باشد. در واقع namespace ها راهی برای کپسوله سازی داده ها و آیتم ها در PHP می باشد.

در دنیای پی اچ پی namespace ها برای حل دو مشکل اصلی طراحی شده اند که عموما کتابخانه ها و برنامه های توسعه دهندگان، با آن ها هنگام برنامه نویسی مواجه می شدند.

در واقع شما، وقتی بخش هایی از کد که برای استفاده مجدد می نویسید، مانند کلاس ها و تابع ها را، می سازید، عموما با دو مشکل زیر مواجه می گردید:

- تداخل نام ها یا همان name collision بین کدهایی که شما ساخته اید با کلاس ها، توابع، ثوابت PHP یا با کلاس ها، توابع یا ثابت های یک برنامه third-party
- ایجاد اشکال ددر هنگام کوتاه کردن) استفاده از (alias نام های بلند یا همان Extra_Long_Namesکه باعث افزایش خوانایی یا readability کد های شما خواهد شد.

برای جلوگیری از name collision یا همان تداخل نام برای class ها، functionها و constantهای هم نام، از مفهوم namespace در PHP استفاده می شود.

در واقع namespace ها راهی را برای گروه بندی کلاس ها، interfaceها، تابع ها و ثابت های مرتبط را به ما می دهند.
در صورتی که دو کلاس یا متد (یا کلاس و متد) داشته باشیم که دارای نام های یکسان باشند، برای جلوگیری از تداخل آن ها باید از namespace جداگانه در قبل تعریف کلاس، استفاده کنیم.
برای تعریف namespace کافی است پیش از عبارت namespace و سپس نام آن استفاده کنیم. در صورتی که namespace از چند بخش (یعنی از دایرکتوری های تو در تو) تشکیل شده باشد، برای جداسازی آن از علامت \ استفاده می کنیم.
برای صدا کردن متد ها یا ساخت اشیا از کلاس دارایnamespace ، می باید نام namespace به همراه علامت بک اسلش فراخوانی شده و بعد از آن، نام متد آورده شود

مفهوم Trait در PHP

زبان هایی چون C++ و پایتون، زبان هایی هستند که از وراثت چندگانه یا همان Multi Inheritanceپشتیبانی می کنند. اما زبان PHP از وراثت چندگانه پشتیبانی نمی کند. برای رفع این مشکل و پیاده سازی مکانیزمی برای استفاده از مجدد از کدهای تکراری در زبان single inheritanceاز مفهوم trait استفاده می شود.
درواقع trait محدودیت های موجود در وراثت یگانه را برداشته و توسعه دهنده می تواند آزادانه از چندین کلاس مختلف که در ساختار سلسه مراتبی کلاسی مختلف قرار دارند، در کلاس های خود استفاده کند.
در واقع مفهوم trait باعث کم شدن پیچیدپی های و جلوگیری از مشکلات رایج مربوط با وراثت چندگانه و Mixin ها می گردد.
traitها بسیار شبیه کلاس هستند، با این تفاوت که از آن ها نمی تواند ارث برد اما می توان از آن ها در کلاس ها به راحتی استفاده کرد. در واقع به نوعی می توان در جاهایی که نیاز داریم به جای مفهوم is-a در شی گرایی از مفهوم has-a یا همان composition استفاده کنیم، از traitاستفاده می کنیم.

برای استفاده از trait کافی به جای کلمه کلیدی class از کلمه کلیدی traitاستفاده کنیم.
همچنین برای استفادهtrait ، کافی است از کلمه کلیدی useدر داخل کلاس استفاده نماییم.

کامنت گذاری با استفاده از DocBlocks در PHP

برای کامنت گذاری کلاس ها در PHP می توانیم از استایل DocBlock استفاده کنیم. بسیاری از IDE های پر استفاده مانند PhpStormیا Eclipseیا Netbeansاز این روش برای تولید خودکار کامنت در کلاس ها، استفاده می کنند.
قدرت واقعی DocBlock ها در توانایی استفاده از tag ها می باشد که با علامت @ شروع می شوند. به این وسیله برنامه نویس می تواند اطلاعات اضافی در مورد بخش های مختلف کلاس را در کد خود، نشانه گذاری و تشریح نمایید.
پر استفاده ترین تگ ها عبارتند:

- author@
- copyright@
- license@
- var@
- param@
- return@

منبع: www.php.net

تهیه شده توسط میلاد سامانی

برای ارتباط با من:

Eng.milad.samani@gmail.com

&

www.miladsamani.ir