Ярмарка Мастеров
livemaster.ru

PHP - Объектно-ориентированное программирование

Даниил Шевчук
Руководитель продуктовой разработки

Содержание

Ярмарка Мастеров
livemaster.ru

- Основы ООП
- Создание классов и объектов (class,__construct)
- Инкапсуляция (public, private, protected)
- Наследование (extends)
- Полиморфизм (abstract, interface)

Основы ООП

Стратегию ООП лучше всего описать как смещение приоритетов в процессе программирования от функциональности приложения к структурам данных. Это позволяет программисту моделировать в создаваемых приложениях реальные объекты и ситуации.

В качестве примера возьмем автомобиль. У него есть колеса, цвет, вид кузова, объем двигателя и так далее. Кроме того, водитель может отдавать ему команды: ехать, остановится, повернуть направо, налево и тп.

Можно говорить о том, что существует некоторый класс автомобилей, обладающий общими свойствами (у всех есть колеса и всем им можно отдавать команды).

Конкретный автомобиль, стоящий на улице - это представитель этого класса, или, другими словами, объект этого класса. У всех объектов этого класса есть свойства: количество колес, цвет, вид кузова и методы: ехать, остановится, повернуть направо, налево.

Создание классов

Перенесем пример с предыдущего слайда на PHP. В PHP класс создается с помощью ключевого слова class, за которым следует название этого класса. И укажем что будет автомобиль будет иметь свойство для цвета и свойство для количества топлива.

```
class Car

{

    public $color; // цвет автомобиля

    public $fuel; // количество топлива

}
```

Создание классов

Давайте теперь сделаем методы нашего класса. В PHP методы, подобно обычным функциям, объявляются с помощью ключевого слова function, перед которым пишется ключевое слово public.

Как уже упоминалось выше, наш автомобиль может ехать, может поворачивать, может останавливаться. Сделаем соответствующие методы в нашем классе:

```
public function go()

{

   // какой-то PHP код

}
```

Объект класса

Мы с вами сделали чертеж нашего автомобиля. Теперь нужно отправится на завод и сделать объект этого класса (то есть конкретный автомобиль).

В PHP это делается с помощью ключевого слова new, после которого пишется имя класса:

```
$myCar = new Car; // командуем заводу сделать автомобиль

// Устанавливаем свойства объекта:

$myCar->color = 'red'; // красим в красный цвет

$myCar->fuel = 50; // заливаем топливо
```

Объект класса

Теперь мы можем управлять созданным автомобилем

```
$myCar->go();
```

Конструктор объекта

```
public function __construct($color, $fuel)
{
}
$car = new Car('blue', 25);
```

Инкапсуляция

Инкапсуляция — свойство языка программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента (что у него внутри?), а взаимодействовать с ним посредством предоставляемого интерфейса.

Мы можем управлять доступом к свойствам и методам класса. Элементы можно объявлять как public (общедоступные), protected (защищенные) и private (закрытые). Рассмотрим разницу между ними:

К **public** (общедоступным) свойствам и методам, можно получить доступ из любого контекста.

К **protected** (защищенным) свойствам и методам можно получить доступ либо из содержащего их класса, либо из его подкласса. Никакому внешнему коду доступ к ним не предоставляется.

Вы можете сделать данные класса недоступными для вызывающей программы с помощью ключевого слова **private** (закрытые). К таким свойствам и методам можно получить доступ только из того класса, в котором они объявлены. Даже подклассы данного класса не имеют доступа к таким данным.

Наследование

Наследование — это механизм позволяющий описать новый класс на основе уже существующего (родительского),при этом свойства и функциональность родительского класса заимствуются новым классом. Наследование реализуется с помощью ключевого слова extends.

```
class Truck extends Car

{

    public function load()

    {

        echo “Груз в машине”;

    }

}
```

Самоподготовка - Полиморфизм

Полиморфизм – это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач.

В PHP полиморфизм реализуется двумя способами: interfaces и abstract classes. Оба имеют свою область применения, и их можно смешивать и сочетать для подгонки в свою иерархию классов.

https://it-black.ru/polimorfizm-klassov-v-php/

https://webformyself.com/polimorfizm-v-obektno-orientirovannom-programmirovanii-na-php/

Самоподготовка - MVC

Model — View — Controller. Шаблоне проектирования, который построен на основании принципа сохранения представления данных. Согласно этому принципу, данные хранятся отдельно от методов, взаимодействующих с этими данными.

https://ru.hexlet.io/courses/php-mvc/lessons/mvc/theory_unit

https://otus.ru/nest/post/1847/

https://geekspace.info/notes/13

Задание

1. Создайте класс User с приватным свойствами name и email, а также методами getName и getEmail которые возвращают соответствующие свойства.

2. Создайте два дочерних класс Покупатель (Buyer) и Продавец (Seller), которые расширяют родительский класс методами buy() и sell() соответственно.

Ярмарка Мастеров
livemaster.ru

Спасибо за внимание

Вопросы?