STEP IT ACADEMY

Работа с ООП

СОЗДАНИЕ WEB–ПРИЛОЖЕНИЙ, ИСПОЛНЯЕМЫХ НА СТОРОНЕ СЕРВЕРА ПРИ ПОМОЩИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP, СУБД MYSQL И ТЕХНОЛОГИИ AJAX

МОДУЛЬ 02. ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА PHP

Содержание

STEP IT ACADEMY

1. Что такое ООП?

ОПРЕДЕЛЕНИЕ И ПРОЦЕДУРНЫЙ ПОДХОД

Определение

Объектно-ориентированное программирование — это подход, помогающий разрабатывать более сложные приложения, сохраняя при этом простоту ориентации в коде для продолжения разработки.

Людям проще воспринимать окружающий мир в виде с помощью взаимодействий его частей друг с другой и разделять их на виды или классы.

Процедурный подход

Без использования ООП применялся *процедурный* подход: программы представляли собой набор функций.

Такой способ разработки для простых и небольших программ. Но чем больше размер кода внутри функции и количество таких функций, тем сложнее их редактировать и поддерживать между ними логическую связь.

STEP IT ACADEMY

2. Класс

ОПРЕДЕЛЕНИЕ, ПРАВИЛА И СИНТАКСИС

Определения

В ООП существует понятие **класса** для понятий из реального мира с описанием шаблона (**свойств**) и функциональных возможностей или **методов** — функций и процедур, которые принадлежат классу.

С технической точки зрения, **объект** — это экземпляр класса, и вы можете создать несколько экземпляров одного и того же класса, а сам класс — это шаблон.

Правила при создании

- Классам следует давать описательные имена.
- По возможности избегайте использования сокращений.
- Каждое слово в имени класса следует писать с заглавной буквы, без использования символа «_».
- Имя класса не должно быть зарезервированным словом.
- Каждый класс следует хранить в отдельном файле, который должен называться, как класс.

Синтаксис

```
class User {
    function methodExample() {
        // method code
    }
}
```

Экземпляр класса

Для создания экземпляра класса используется переменная, которой присваивается название класса с использованием конструкции new:

```
$user = new User;
```

Оператор объекта

После, создания методы и свойства класса могут быть доступны экземпляру через оператор объекта — символы ->:

```
$user->methodExample();
```

Ссылка на объект

Также с помощью оператора объекта и конструкции `$this` внутри метода создаются переменные класса:

```
function methodExample() {
    $this->varName = "value";
}
```

3. Конструктор

ПРИНЦИП РАБОТЫ И ПРИМЕР

Принцип работы

Конструктор — необязательный метод, который вызывается при создании экземпляра.

PHP ищет метод `__construct()` и автоматически вызывает.

Также конструктор может принимать аргументы, что значительно упрощает работу с классами

Пример

```
class User {
    function __construct(String $string) {
        echo $string;
    }
}
```

Использование

Такой код при создании экземпляра класса `User` будет требовать передачи строки и выводить её на страницу:

```
$user = new User("Hello, World!");
```

Следует учесть, что аргументы, принимаемые конструкторов, необходимо указывать при каждом вызове класса.

Если у класса нет конструктора, или конструктор не имеет обязательных параметров, скобки после имени можно не писать.

STEP IT ACADEMY

4. Принципы

ВСЕ ЧЕТЫРЕ ВИДА

Примеры

Разработка с помощью объектно-ориентированного подхода строится на четырёх основных принципах:

- Абстракция
- Инкапсуляция
- Полиморфизм
- Наследование

1. Абстракция

Абстракция подразумевает выделение общих характеристик объектов без лишней информации.

Главная особенность абстракции заключается в отсутствии излишней детализации. Например, при использовании кофемашины используется вода, зёрна и выбирается тип кофе.

Пример абстракции

```
class CoffeeMachine {
    function PourWater() { // Заливка воды }
    function AddBeans() { // Засыпка зёрен }
    function BrewCoffee() {
        // Заварка кофе. При этом, содержание метода для разных
        // кофемашин будет отличаться
    }
}
```

2. Инкапсуляция

Инкапсуляция позволяет вводить ограничения на доступ к участкам кода. Например, можно сделать метод, доступный только для внутреннего использования в классе с помощью *модификаторов доступа*, которые и определяют наличие ограничений.

Модификаторы доступа

Для определения доступности существует понятия модификаторов доступа, которые и определяют наличие ограничений. Их существует три вида:

- `public` — доступ к свойствам и методам из любого места
- `protected` — доступ к родительскому и наследуемому классу
- `private` — доступ только из класса, в котором объявлен сам элемент.

Использование модификаторов

Модификатор по умолчанию — `public`. У свойств значения модификатора по умолчанию нет, поэтому он может быть задан при объявлении переменных в теле класса.

Использование модификаторов

```
class User {
    public $id = null;
    public function __construct(Int $id) {
        $this->id = $id;
    }
}
```

3. Наследование

Наследование позволяет создать класс на основе существующего, используя уже готовые методы или даже конструктор. Так один класс (*parent*) может лежать в основе другого класса (*child*).

Для создания класса-наследника используется ключевое слово `extends`, указывая, что новый класс будет являться расширением основного.

Преимущества наследования

- Наследник может переопределять родительские методы и свойства, реализовывать собственные.
- Общий функционал реализуется в родителе, а все *подклассы* наследуют его.
- Использование наследования позволяет разбивать большие и сложные классы на более мелкие и управляемые.

4. Полиморфизм

Главная особенность полиморфизма — это возможность разным конструкциям выполнять одни и те же действия. При этом, различия конструкций и их устройство значения не имеют.

Дополнительная литература

1. Документация по PHP: https://www.php.net
2. Классы и объекты : https://www.php.net/manual/ru/language.oop5.php
3. Область видимости: https://www.php.net/manual/ru/language.oop5.visibility.php