web разработка

ООП въведение

Съдържание

- Обектно ориентирано програмиране -
  - същност
  - OOP vs. процедурно програмиране
    - разлики
    - преимущества
- Класове и обекти - същност, дефиниране
- Методи и свойства - същност и достъп
- конструктор

Обектно ориентирано програмиране
Същност

ООП - същност - 1

Уеб приложение, разработено с процедурно програмиране -

Предаване на информация/данни - от една променлива на друга, от една функция на друга…

Кодът е неразбираем и недобре организиран.

Едни и същи парчета код се повтарят многократно.

ООП - същност - 2

Уеб приложение, разработено с ООП -

Уеб-приложението е взаимодействие между различни ОБЕКТИ

Потребители - администратор, клиент …

Страници - начална, страница визуализираща продукт, страница визуализиращ списък от продукти ...

Всеки от т.нар. ОБЕКТИ - изпълнява определена задача

Сравнете потребители и страници

Сравнете Администратор, Клиент, Друг Тип потребител

Като функции и характеристкики, които притежава по отношение на уеб приложението, което разработваме

ООП - същност - 3

Уеб приложение, разработено с ООП -

Преизползване на кода - описанието на Администратор и Клиент ще има общи и различни функции и характеристики.

Да ги изброим ….

Модулност и разширяемост на кода.

Добавяме нови функционалности без да засягаме наличните.

В уеб приложението може да добавим

- *Опция за регистрация*
- *Страници с достъп само след регристрация*
- *Достъп за определен вид потребители*

След като е било напълно работещо и без тези нововъведения и без те да попречат на началната функционалност ...

ООП - същност - 4

Да обобщим

Процедурно програмиране -

неголеми и несложни проекти

ООП -

Сложни проекти

Разработвани от екипи от програмисти

Всеки може да разработва независимо различна функционалност

Потребители

Коментари

Качване на файлове

Други ...

Клас

Клас

Класът е метод за описание на дадена същност.

Същност - автомобил /по принцип какво е абтомобил?/

- Потребител /в нашето уеб приложение/
- Страница /от нашето уеб приложение/

Клас - 2

Класът е метод за описание на дадена същност

ЧЕРТЕЖ / ШАБЛОН / ДРУГ СИНОНИМ ...

Нещо, което описва как изглежда

и какво може да прави

Един АВТОМОБИЛ

ПОТРЕБИТЕЛ

СТРАНИЦА

По принцип /без да се обвързваме с конкретния МОЙ АВТОМОБИЛ, АДМИНИСТРАТОРЪТ ПЕШО или ПОТРЕБИТЕЛЯТ ИВАН/

Опишете класовете Автомобил, Потребител, Страница

Обект

Обект

След като сме изготвили чертежа -

КЛАС АВТОМОБИЛ

```
class Automobile {
}
```

КЛАС ПОТРЕБИТЕЛ

```
class User {
}
```

Обект - 2

След като сме изготвили чертежа - по него произвеждаме автомобили, създаваме потребители …

Автомобилите - могат да са в различен цвят, да имат различни …..

Потребителите - имат различни потребителски имена, пароли …..

Заради разликата в свойствата не променяме чертежа - КЛАСЪТ.

Обект - 3

По чертежа/КЛАСЪТ - създаваме обект. Това е конкретния автомобил, който е създаден с

Максимална скорост - … км/ч

Цвят - син

Брой врати - 4

…..

Обект - 4

Или ПОТРЕБИТЕЛ

Потребителско име - Пешо

Парола - secret

Права - admin

Обект - 5

```
class Automobile {
        class Page {

}
                }
```

```
class User {


        }
```

```
$car = new Automobile()
new Page()
```

```
$pesho = new User()
```

```
$index =
```

Обект - 6

Дефиницията на всеки клас се запазва във файл със същото име.

Един клас - един файл.

```
class Page {

}
```

Page.php/page.php

Свойства/Properties

Свойства

Свойствата са променливи.

```
class  Page {

        public $header = ‘HEADER’;
        public $content;
        public $footer;
}
```

public - спесификатор за достъп

Свойства - 2

Достъп до свойствата на обекта

```
$index = new Page();

echo $index->header;
            //HEADER

$index->header = ‘INDEX PAGE HEADER’;
echo $index->header;        //INDEX PAGE HEADER
```

Класът не е променен. Променяме стойността на свойството header на самия обект.

Свойства - 3

Да зададем стойност на свойствата

- content
- footer

И да ги отпечатаме на екрана ...

Методи/Methods

Методи

Методите са функции, описани в рамките на класа.
Също имат спесификатори за достъп.

```
public function view_header() {

        echo ‘Hello’;

}
```

Методи - 2

Достъп до метода

```
class Page {

public $header;

        public function view_header($header){
                        echo $header;
        }
}
$account = new Page();
$account->view_header('ACCOUNT HEADER');//ACCOUNT HEADER
```

$this

$this

```
class Page {

public $header;

    public function view_header(){
                echo $this->header;
    }
}
$account = new Page();
$account->header = 'ACCOUNT HEADER';
$account->view_header();//ACCOUNT HEADER
```

$this - 2

За да се обърнем към метод или свойство на клас в самия клас използваме $this.
Използване в метод в класа

```
public function view_info($param1, $param2){
        echo $param1. ‘ - ’ . $param2;
}
public fucntion foo() {
        $var1 = 1;
        $var2 = 2;

        $this->view_info($var1, $var2);

        }
```

Спесификатори за достъп

Спесификатори за достъп

Ограничават достъпа до методите и свойствата на класа извън него - за неговите наследници, други класове, ...

PUBLIC

PRIVATE

PROTECTED

Спесификатори за достъп

```
class spec {
Spec();
        public $a;
        protected $b;
        private $c;
}
```

```
$spec = new

echo $spec->b;

echo $spec->c;
```

Спесификатори за достъп

Използват се за да се елиминира случайно /или не/ изменение на методи и свойства.

Пример - връзка към база данни

Чрез спесификаторите за достъп “скриваме” реализацията на класовете /енкапсулация/. Работим с обектите и техните свойства, без да се интересуваме от начина, по който са реализирани класовете.

конструктор/__construct()

конструктор

Обикновен метод, който се **изпълнява в момента на създаване на обект**.

```
public function __construct() {

}
```

```
class Page {
    public $header;

    public function __construct() {
        echo 'NEW PAGE HAS BEEN CREATED';
    }
}
```

конструктор

Обикновен метод, който се **изпълнява в момента на създаване на обект**.

```
public function __construct() {

}
        class Page {
                public $header;

                public function __construct($h) {
                                        $this->header = $h;
                echo $this->header;
                }
            }
```

Конструктор - 3

При създаването на обект, чрез конструктора може да зададем стойност на свойствата му.

```
public function __construct($param) {

        echo $this->property = $param;

}
```

Задаване на стойност на свойството **property.**

```
$index = new Page('something');
echo $index->property; //something
```

Конструктор - 3

Щом сме декларирали, че конструкторът ще приема параметри, при създаването на всеки обект трябва да подаваме нужните параметри.