web разработка

ООП Наследяване

Съдържание

- Наследяване на класовете
- Предефиниране на методи в класовете при наследяване
- isA, hasA - връзки между класовете - демо
- Спесификаторите за достъп при наследяване - демо
- Final

Наследяване на класовете

Class inheritance

Class inheritance

```
class Page {
	public $title;
	public $content;
	public $footer;
	public function __construct($t, $c, $f) {
		$this->title = $t;
		$this->content = $c;
		$this->footer = $f;
	}
	public function render_header(){
		$str = $this->title();
		return $str;
	}
	public function render_title() {
		$str = '<h1>'.$this->title.'</h1>';
		return $str;
	}
	public function render_conent() {
		$str .= '<p>'.$this->content.'</p>';
		return $str;
	}
	public function render_footer() {
		$str .= '<p>'.$this->footer.'</p>';
		return $str;
	}
}
```

Class inheritance - 2

Задача Създайте приложение, което …..

Страниците на приложението са Home, Contacts, About Us, Content
Само Home страницата има слайдер и реклама.
Всички страници имат

- header
- content
- footer

Class inheritance - 3

За да създаваме обекти от клас Home Page, притежаващи описаните свойства, трябва да копираме клас Page и в него да добавим характерните/различните за Home Page свойства.

/Home Page съдържа всички характеристики на останалите страници и в допълнение има слайдер и реклама/.

Class inheritance - 4

Properties

```
class HomePage {

    public $title;
    public $content;                    <= повтарящ се код
    public $footer;
    public $slider;
    public $banner;          <= специфични за класа HomePage свойства
```

Class inheritance - 5

Повторенията в кода се избягват като се използва наследяването на класовете или

```
class HomePage extends Page {

        public $slider;
        public $banner;


                ……
```

$homepage притежава всички свойства, характерни за обектите от клас Page плюс тези, принадлежащи на клас HomePage.

Class inheritance - 6

Methods

Класът HomePage, като наследник на Page, притежава свойствата и МЕТОДИТЕ на Page.
Може да има и свои методи и/или да предефинира или разширява родителските методи/тези на Page/.

Class inheritance - 7

Methods

```
class HomePage extends Page {

        public $slider;
        public $banner;

        public function __construct($h, $c, $f, $s, $b){
                parent::__construct($h, $c, $f);//извикваме задължително родителския конструктор, ако
                $this->slider = $s;                                     //искаме в момента на създаването на обект
                $this->banner = $b;                             //да се придаде стойност на всички свойства
        }

}
```

Class inheritance - 7

Methods

```
class HomePage extends Page {

        public $slider;
        public $banner;

        public function __construct($h, $c, $f, $s, $b){
                parent::__construct($h, $c, $f);          //извикваме задължително родителския конструктор, ако
                $this->slider = $s;                                    //искаме в момента на създаването на обект
                $this->banner = $b;                                    //да се придаде стойност на всички свойства
        }

        public function render_slider() {
                $str .= '<p>'.$this->slider.'</p>';
                return $str;
        }
        public function render_banner() {
                $str .= '<p>'.$this->banner.'</p>';
                return $str;
        }
}
```

Предефиниране на методите в класа
Method overriding

Method overriding

В класът HomePage предефинирахме __construct(). Освен на header, content, footer, той задава стойности и на slider и banner свойствата.

Можем да предефинираме и други методи в класовете наследници

- В наследника създаваме едноименен метод с този в родителския
- Метода ще функционира във формата, дефиниран
  - В родителския клас, ако се извиква от негови обекти
  - В дъщерния клас, ако се извиква от негови обекти.

Method overriding - 2

```
//дефиниция в Page

        public function render_header(){
                $str = $this->title();
                return $str;
        }


//дефиниция в HomePage

        public function render_header(){
                $str = parent::render_header();
                $str .= $this->slider();

                return $str;
        }
```

IsA / HasA
demo

final

final

Когато декларацията на метод започва с ключовата дума **final**, този метод не може да бъде предефиниран в класовете наследници.

```
class Test {

….

 final public function moreTesting() {

      echo "I am a final class and cannot be overriden!";

   }

...

}
```

final

Когато клас е означен като **final** - не може да бъде наследяван.

final

```
final class BaseClass {
   public function test() {
      //
   }

   // Here it doesn't matter if you specify the function as
final or not
   final public function moreTesting() {
      //
   }
}
```

```
class ChildClass extends BaseClass {

}

// Results in Fatal error: Class ChildClass may not inherit
from final class (BaseClass)
```

Access modifiers
Спесификатори за достъп

Access modifiers

more info

1. *public* – class or its members defined with this access modifier will be publicly accessible from anywhere, even from outside the scope of the class.
2. *private* – class members with this keyword will be accessed within the class itself. It protects members from outside class access with the reference of the class instance.
3. *protected* – same as private, except by allowing subclasses to access protected superclass members.
4. *abstract* – This keyword can be used only for PHP classes and its functions. For containing *abstract* functions, a PHP class should be an *abstract* class./to be explained later/
5. *final* – It prevents subclasses to override super class members defined with *final* keyword.