web разработка

ООП Наследяване

Съдържание

Наследяване на класовете
Предефиниране на методи в класовете при наследяване
Полиморфизъм

Part 2
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_body() {
		$str = ‘<h1>’.$this->$title.‘</h1>’;
		$str .= ‘<p>’.$this->$content.‘</p>’;
		$str .= ‘<p>’.$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 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

```
class HomePage {
        public $title;
        public $content;                                    <= повтарящ се код
        public $footer;
        public $slider;
        public $banner;                    <= специфични за класа HomePage свойства
        public function __construct($t, $c, $f) {
                $this->$title = $t;                                  <= повтарящ се код
                $this->$content = $c;
                $this->$footer = $f;
        }
        public function render_body() {
                $str = '<h1>'.$this->$title.'</h1>';
                $str .= '<p>'.$this->$content.'</p>';       <= повтарящ се код
                $str .= '<p>'.$this->$footer.'</p>';

                return $str;
        }
}
```

Class inheritance - 5

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

```
class HomePage extends Page {

        public $slider;
        public $banner;

}

$homepage = new HomePage();

$homepage->slider = ‘Slider’;
$homepage->banner = ‘Banner’;
```

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

Class inheritance - 6

Класът HomePage може да има свой конструктор.

```
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
        }

}
```

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 HomePage
public function render_body() {

        parent::render_body();

        $str .= ‘<p>’.$this->slider.‘</p>’;
        $str .= ‘<p>’.$this->banner.‘</p>’;

        return $str;

}
```

Дали ще извикваме родителския метод при предефинирането му в дъщерния, зависи от логиката на задачата, която решаваме.

Задача

1. Създайте клас наследник ContactPage
2. Опишете свойствата на този клас
3. Опишете негов метод
4. Предефинирайте в него метод на родителския клас

Полиморфизъм

Полиморфизъм

В класа, който разработваме може да използваме методи, дефинирани в други класове.

Полиморфизъм - 2

```
class Poly {
        //записва обекти в масивът $obj
        public $obj =[ ];
                //записва обекти в $obj.
                //Изрично указваме, че обектите трябва да бъдат от клас Page,
                //което включва и неговите наследници
        public function get_obj(Page $page){
        $this->obj[ ] = $page;
        }
        //извежда резултатът от работата на нашия клас
        public function get_result() {
                foreach ($this->obj as $item) {
                echo $item->test();//метод, описан в класът Page
                }
        }
}
```

Полиморфизъм - 3

```
class Page {
public $page_name;
        ...
        public function test() {

                return $this->page_name;

        }
}
$poly = new Poly();

$poly->get_obj($page);

$poly->get_obj($index);

$poly->get_result();
```

```
$page = new Page( …. )//не забравяйте нужните параметри

$page = new Index(.....)//не забравяйте нужните параметри
```

Спесификатори за достъп

Спесификатори за достъп

```
class Page {
public $page_name;
……
}

class HomePage {
…..
}

$home_page = new HomePage()
echo $home_page->page_name; //ще се отпечата името
```

Спесификатори за достъп

```
class Page {
protected $page_name;
……
}

class HomePage {
…..
}

$home_page = new HomePage()
echo $home_page->page_name;          //Fatal error: Uncaught Error: Cannot access protected property
                                     //HomePage::$page_name
```

Спесификатори за достъп

```
class Page {
protected $page_name;
……
}

class HomePage {
        public function get_pagename(){
                echo $this->page_name;
        }
}

$home_page = new HomePage()
echo $home_page->page_name;
$home_page->get_pagename();
```

Спесификатори за достъп

```
class Page {
private $page_name;
……
}

class HomePage {
        public function get_pagename(){
                echo $this->page_name;
        }
}

$home_page = new HomePage()
echo $home_page->page_name;
$home_page->get_pagename(); //Undefined property: HomePage::$page_name
```

Setter
Getter

Setter Getter

get_name()/set_name().

Тези методи следват общата за много езици конвенция за ООП, за задаване/достъпване на свойствата на един клас.

Според конвенцията за имената, имената на getter и setter методите трябва да отговарят на имената на свойствата.

Предназначение – позволяваме задаване на стойност на свойството или четенето му извън класа.

```
class Person {

public $name;

function set_name($new_name) {

$this->name = $new_name;

}

function get_name() {

return $this->name;

}

}
```

Setter Getter

__set() – записване на данни /недостъпни свойства/

```
class PropertyTest {
    private $data;
    public function __set($name, $value) {
        echo "Setting '$name' to '$value'\n";
        $this->data = $value;
    }
…..
}
```

Setter Getter

__get() – четене на данни /недостъпни свойства/

```
class PropertyTest {
    private $data;
…
    public function __get($name)   {
        echo "Getting '$name'\n";
        return $this->$name;
    }
}
```