Vzor Singleton

Návrhový vzor Singleton umožňuje vytvářet objekty, u nichž je zaručena jejich unikátnost. Znamená to, že díky tomuto vzoru je možné objekt volat vždy pouze jednou.

Vzor **Singleton** definuje třídu, která má pouze jednu instanci.

Představme si například volání obsahu určitého souboru. Konstruktor by vždy otvíral soubor, přestože je už otevřený. Použijeme-li však pro práci se souborem třídu odvozenou podle vzoru **Singleton**, snadno určíme, zda je soubor otevřen a zda s ním můžeme pracovat.

Podívejme se na podobný příklad:

```
<?php
  class Singleton {
    private static $instance = false;
    public $vlastnost;
    private function __construct(){}
    public static function getInstance(){
      if(self::$instance === false){
        self::$instance = new Singleton;
      }
      return self::$instance;
    }
  }

  $a = Singleton::getInstance();
  $b = Singleton::getInstance();
  $a->vlastnost = "Vypiš vlastnost 'vlastnost'";
  print $b->vlastnost;
?>
```

Skript volá třídu Singleton, která obsahuje statickou vlastnost `$instance` a veřejnou vlastnost `$vlastnost`. Privátní konstruktor zakazuje vytváření instancí mimo třídu. Statická metoda `getInstance()` testuje, zda je instance vytvořená - pokud ano, vracíme instanci, pokud ne, instanci vytváříme. Mimo třídu pak voláme metodu `getInstance()` dvakrát, poprvé vytváříme instanci, podruhé ji vracíme. Dále přiřazujeme hodnotu vlastnosti `$vlastnost`, kterou pak zobrazujeme v prohlížeči.

Pozn. red.: Zdůrazňujeme, že tento text se týká PHP 5.

zpracoval: Mrozek, Jakub (16. 11. 2006)

Komentáře k článku

Od: heltone
IP: 90.176.60.6

Datum: 2007-06-20 23:28:35

Zdravim, nevim jestli to patri primo sem, ale mam takovy dotaz:

mam treba nejaky:
<? index.php
require "necorequire.php";

$string = " blabol "
$obj->fce_pridej_k_vypisuhtml($string);
?>

v tom
<? necorequire.php
$obj_vypishtml = vypishtml::getInstance();
$obj_dalsiobjekt = new dalsiobjekt;
?>

ta trida
<? vypishtml.class.php
class vypishtml {

private static $instance = false;
private $celyhtml;

public static function getInstance(){
if(self::$instance === false){
self::$instance = new vypishtml();
}
return self::$instance;
}

fce_pridej_k_vypisuhtml($str) {
$this->$celyhtml.=$str;
}

fce_vypishtml() {
print $this->celyhtml;
}
}
?>

proste ten skript v i
<? dalsiobjekt.class.php
$string = " Ma proste neco pripsat ";

$obj_vypishtml->fce_pripoj_k_vypisu($st…
?>

vlastne jakykoli objekt muze volat $obj_vypishtml i vsechny jeho metody, jenze !?

Kdyz si $obj_vypishtml->fce_pripoj_k_html($stri… zavolam v index.php vsechno je okay, funguje (ackoli jeho instanci tvorim az v necorequire.php) a v index.php metodu volam pozdeji nez v dalsich objektech/jinych.php.

Kdyz vsak chci: $obj_vypishtml->fce_pripoj_k_html($stri…

Pokud ho vsak chci volat nekde v dalsiobjekt.class.php podobne jako v index.php uz mi to nejde, musim predtim zase volat

$obj_vypishtml = vypishtml::getInstance();

potom az

$obj_vypishtml->fce_pripoj_k_html($stri…

pak to kupodivu funguje :]

Muzu jeste udelat globals $celyhtml; nebo jeste vyjetejc $_SESSION['celyhtml'].=$string;

Na konec jeste mala podotazka, je nejaky podstatny rozdil mezi v pouziti zavorek at uz jakkoli (kompatibilita ...):
$obj1 = new tridanejaka;

vs

$obj2 = new tridanejaka();

ps: predpokladam, ze nevyzaduji zadne parametry pro konstruktor;

Diky za komenty ....

Od: Tomor
IP: 213.192.18.14
Datum: 2008-05-01 14:20:00

To co popisuješ je správné a logické chování proměnných v PHP.
Když si v indexu vytvoříš proměnnou. Uvnitř třídy do ni můžeš jedině pres $GLOBALS['promena'], nebo si ji poslat dovnitř v konstruktoru.

Metody singletonu se standardně volají tímto způsobem(odkudkoliv - i z jiné třídy):

vypishtml::getInstance()->fce_pripoj_k_…

Odpověď na podotázku:
netuším zda kompatibilita já používám php5 a všude new Trida() - se zavorkami

---

Od: Megaloman
IP: 147.229.216.182
Datum: 2008-07-14 10:50:23

Pokud vytvářítе spousty tříd podle návrhového vzoru Singleton, nebo často měníte jejich název a jste líní přepisovat jejich název v metodě getInstance(), můžete to obejít následovně:

```
public static function getInstance()
{
if (self::$instance === false)
{
```

```
$class = __CLASS__;
self::$instance = new $class();
}
return self::$instance;
}
```

Přímý zápis
... = new __CLASS__();
nelze použít.

Od: ktx
IP: 80.188.207.175
Datum: 2008-07-29 12:25:53

Re: Megaloman
Možná lepší než ten trik s __CLASS__ je hodit new self();

To self je vůbec kouzelná věc, jdou pak dělat věci jako:

```
public static function getInstance() {
   if(self::$instance instanceof self) return self::$instance;
   return self::$instance = new self();
}
```

... ale to už moc prasárna i na mě. :)

Přidejte svůj názor!

. ISSN 1212-8651 .