Programación web con PHP

Patrones: Decorator

Temario

Tipo

Estructural (Nivel objetos)

Propósito

Permite añadir funcionalidades de manera dinámica funcionando como una alternativa mas flexible a la herencia cuando se trata de funcionalidad.

Utilización

- Cuando se quiere añadir responsabilidades a un objeto de manera dinámica e independiente de otros objetos
- Cuando no es posible extender la funcionalidad por herencia ya que no se puede saber por anticipado el tipo y la cantidad.

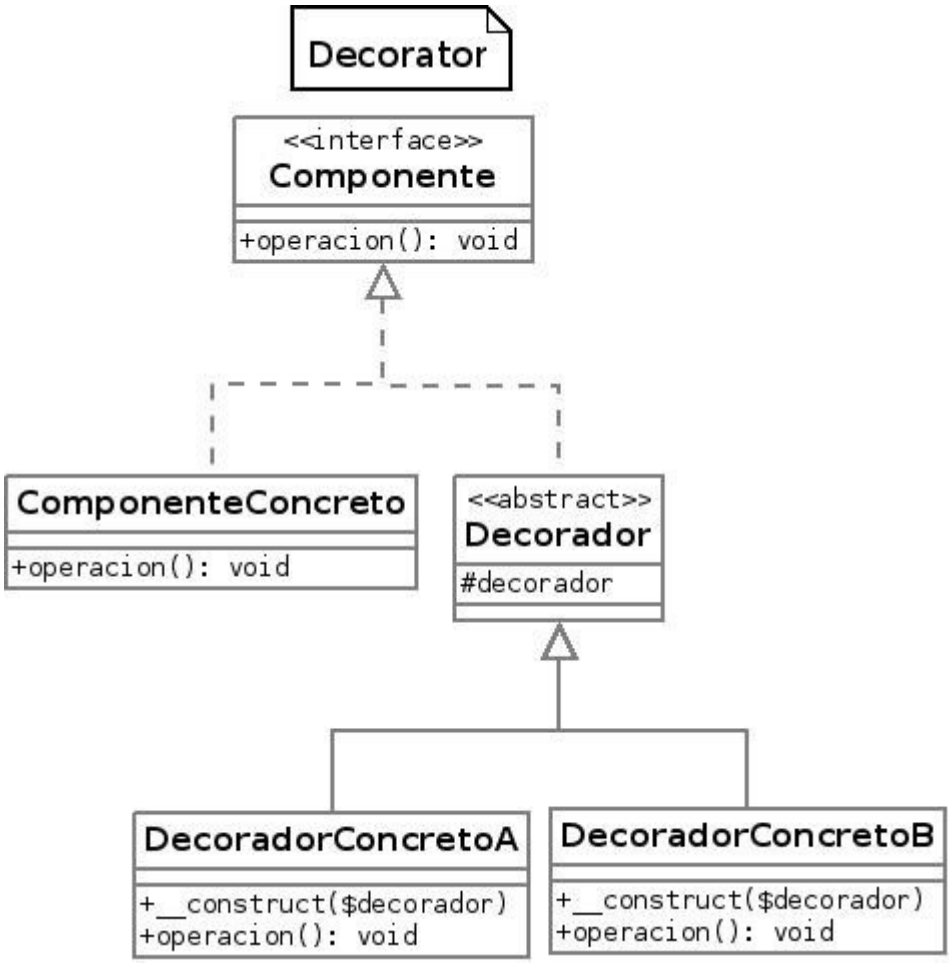
Ventajas

- Mayor flexibilidad que con herencia permitiendo agregar una funcionalidad mas de una vez

Desventajas

- Aumenta el numero de clases pequeñas lo que hace mas difícil el aprendizaje del sistema

Estructura



¿Cómo funciona?

- Definimos el Componente, que será una interfaz que contiene las definiciones de los métodos utilizados por los Componentes Concretos.
- Definimos el Componente concreto que implementará los métodos de la interfaz.
- Definimos el Decorador que tendrá como atributo protegido un decorador .
- Definimos tantos decoradores concretos como sea necesario que extenderán del decorador, en su constructor setearan el atributo protegido decorador al decorador pasado por parámetro y cada uno de sus métodos ejecutarán la operación del decorador guardado en el atributo.
- En el contexto creamos el decorador concreto que necesitemos pasándole como parámetro el componente concreto, en el caso que queramos aplicar mas de un decorador concreto pasamos el segundo decorador como parámetro y dentro de este el componente concreto
- Al ejecutar una operación se ejecutaran las operaciones de todos los decoradores concretos implementados

Código

Componente

```
//Componente
interface Componente
{
	public function operacion();
}
//Componente Concreto
class ComponenteConcreto implements Componente
{
	public function operacion()
	{
		echo "Ejecutando operación<br>";
	}

}
```

Componente Concreto

```
<?php
//Componente Concreto
class ComponenteConcreto implements Componente
{
	public function operacion()
	{
		echo "Ejecutando operación<br>";
	}

}
```

Decorador

```
<?php
//Decorador
abstract class Decorador implements Componente {

	protected $decorador;
}
```

Decorador Concreto A

```php
<?php
//Decorador Concreto
class DecoradorConcretoA extends Decorador {

	function __construct($decorador)
	{
		$this>decorador=$decorador;
	}
	public function operacion() {
		$this>decorador>operacion();
		echo "Operación de Decorador Concreto A<br>";
	}

}
```

Decorador Concreto B

```php
<?php
//Decorador Concreto
class DecoradorConcretoB extends Decorador {

	 function __construct($decorador)
	{
		$this>decorador=$decorador;
	}

	public function operacion() {
		$this>decorador>operacion();
		echo"Operación de Decorador Concreto B<br>";
	}

}
```

index.php

```
<?php
//Creo un componente con el decorador A
$componente1=new DecoradorConcretoA(new ComponenteConcreto());
$componente1>operacion();//Ejecutando operación Operación de Decorador
Concreto A
echo "<br><br>";
//Creo un componente con el decorador B
$componente2=new DecoradorConcretoB(new ComponenteConcreto());
$componente2>operacion();//Ejecutando operación Operación de Decorador
Concreto B
echo "<br><br>";
//Creo un componente con el decorador A y el decorador B
$componente3=new DecoradorConcretoB(new DecoradorConcretoA(new
ComponenteConcreto()));
$componente3>operacion();//Ejecutando operación Operación de Decorador
Concreto A Operación de Decorador Concreto B
echo "<br><br>";
```