Design Pattern em PHP

Prof.: Alisson G. Chiquitto
chiquitto@unipar.br

Design Pattern

“(...) um Design Pattern é uma solução reusável para problemas que ocorrem com frequência (...)” - Wikipédia

Factory Pattern

- O padrão Factory fornece uma interface para a criação de famílias de objetos correlatos ou dependentes sem a necessidade de especificar a classe concreta destes objetos.

Factory Method

```php
interface CarFactory {
    public function makeCar();
}
interface Car {
    public function getType();
}
class SedanFactory implements CarFactory {
    public function makeCar() {
        return new Sedan();
    }
}
class Sedan implements Car {
    public function getType() {
        return 'Sedan';
    }
}

$factory = new SedanFactory();
$car = $factory->makeCar();
print $car->getType();
```

Factory Method

```php
class ConexaoMysql extends PDO {
    public function __construct() {
        parent::__construct('STRING_MYSQL', 'root', 'teste');
    }
}
class ConexaoSqlite extends PDO {
    public function __construct() {
        parent::__construct('STRING_SQLITE');
    }
}
class Conexao {
    public static function factory($option) {
        switch ($option) {
            case 'mysql':
                return new ConexaoMysql();
                break;
            case 'sqlite':
                return new ConexaoSqlite();
                break;
            default:
                throw new Exception('Option desconhecido');
        }
    }
}
$con = Conexao::factory('mysql');
```

Singleton Pattern

- Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto.

Singleton Pattern

```php
class Singleton {

    private static $_instance;

    // Evita que a classe seja instanciada publicamente
    private function __construct() {

    }

    // Evita que a classe seja clonada
    private function __clone() {

    }

    public static function getInstance() {
        if (!isset(self::$_instance)) {
            self::$_instance = new self;
        }
        return self::$_instance;
    }

}
```

Singleton Pattern

```php
class Exemplo {
    // Guarda uma instância da classe
    private static $instance;

    // Um construtor privado;
    // previne a criação direta do objeto
    private function __construct() {
        echo 'Sou um construtor';
    }

    // O método singleton
    public static function singleton() {
        if (null === self::$instance) {
            self::$instance = new Exemplo();
        }
        return self::$instance;
    }

    // Previne que o usuário clone a instância
    public function __clone() {
        throw new Exception('Clone is not allowed.');
    }

}
```

Referencias

- Software Design Pattern. Wikipédia. <http://en.wikipedia.org/wiki/Software_design_pattern>
- Design Patterns - o padrão Factory. Macoratti. <http://www.macoratti.net/vbn5_dp1.htm>
- PHP: Patterns – Manual <http://php.net/manual/pt_BR/language.oop5.patterns.php>