# PHP Coding Standards

**v0.6 beta**

## Walker de Alencar Oliveira

contato@walkeralencar.com





# Introdução

- Como escrever um código organizado, bem estruturado e documentado. Melhorando visibilidade e facilitando futuras manutenções e implementações.
- Um bom padrão de codificação é importante em qualquer projeto de desenvolvimento, principalmente quando envolve vários desenvolvedores.
- Assegurar a Alta Qualidade o código, diminuir bugs.

# Formatando Arquivos PHP

- Identação
  - 4 espaços
  - Tab no tamanho de 4 espaços.
- Tamanho máximo da Linha
  - 80 caracteres, mas é aceitável até 120 caracteres.
- Término de linha
  - Linhas devem terminar apenas com Linefeeds (LF) – [\n]
  - Não usar Carriage Return (CR) – [\r], padrão Macintosh
  - Não usar a combinação (CR)(LF) – [\r\n], padrão Windows

```
<?php
/**
 * [Descrição do arquivo].
 *
 * [mais informações precisa ter 1 [ENTER] para definir novo parágrafo]
 *
 * [pode usar quantas linhas forem necessárias]
 * [linhas logo abaixo como esta, são consideradas mesmo parágrafo]
 *
 * @package    [Nome do pacote de Classes, ou do sistema]
 * @category   [Categoria a que o arquivo pertence]
 * @name       [Apelido para o arquivo]
 * @author     [nome do autor] <[e-mail do autor]>
 * @copyright  [Informações de Direitos de Cópia]
 * @license    [link da licença] [Nome da licença]
 * @link       [link de onde pode ser encontrado esse arquivo]
 * @version    [Versão atual do arquivo]
 * @since      [Arquivo existe desde: Data ou Versao]
 */
...
```

# Nomenclaturas

- **Padrão CamelCase**
  - ***Definição***
    - É a denominação em inglês para a prática de escrever palavras compostas ou frases, onde cada palavra é iniciada com Maiúsculas, e unida sem espaços.
    - É um padrão largamente utilizado em diversas linguagens de programação, como Java, Ruby e Python, principalmente nas definições de Classes e Objetos. (Fonte: wikipedia.com)
  - ***SubDivisões***
    - lowerCamelCase - iniciado por letra Minúscula
      - iPod, iMac
    - UpperCamelCase - iniciado por letra Maiúscula
      - GameCube, OpenOffice.org , StarCraft
- UpperCamelCase
  - Classes
    - Uploads, FileUploads, ImageFileUploads
- lowerCamelCase
  - Variáveis
    - $tmpQry, $objUpload, $arrUF, $iCount
  - Propriedades
    - $this ->tableName, $this ->fieldId, $this ->fields
  - Funções e Métodos
    - $this->getName(), $this ->setFields(), $this ->getById()
- UPPERCASE e _
  - Constantes
    - NOT_EXIST, OVERFLOW_MAX_SIZE, DB_HOST

# Estilo de Código (Coding Style)

- Demarcação de Código PHP
  - Não usar short_tags(**<?** e **<?=**)
  - Usar tags completas (**<?php e <?php echo** )
- Strings Literais
  - Se a string não contiver variáveis de substituição, deve-se usar aspas simples.
    - `$tmpStr = 'Exemplo de String';`
- Strings Literais com Apóstrofos
  - Pode-se usar aspas simples, mas é recomendado o uso de aspas duplas para evitar slashes [\]
    - `$tmpSql = "SELECT id, nome FROM cliente WHERE name = 'Walker' ";`
- Substituição de Variáveis
  - Usar aspas duplas.
  - Pode-se usar de 2 formas:
    - `$tmpStr = "Exemplo de String com $variavel ";`
    - `$tmpStr = "Exemplo de String com {$variavel} ";`
  - Por consistência não se recomenda usar a forma abaixo:
    - `$tmpStr = "Exemplo de String com ${variavel} ";`
- Concatenação de Strings
  - Usar espaço antes e depois do operador ".", melhorando assim a visibilidade.
    - `$tmpStr = 'Exemplo ' . $de . " String com {$variavel} "; // Exemplo`
    - `$tmpStr = "Exemplo {$de} String com {$variavel} "; //Recomendado`
  - Quando concatenar mais de uma string longa, alinhe o operador "." abaixo do operador "=".

```
$tmpSql = 'SELECT id,nome '
        . 'FROM cliente '
        . "WHERE name = 'Walker' ";
```

## Classes

- Nomear em UpperCamelCase.
- As chaves "{" e "}" virão na linha abaixo do nome da Classe.
- Toda classe deve ter um bloco de documentação em conformidade com o Padrão do PHPDocumentor.
- Qualquer código dentro da classe precisa ser identado com quatro espaços.
- Só é permitida uma classe por arquivo PHP.

```php
/**
 * Bloco de Documentação
 */
class ClasseExemplo
{
    /**
     * @access private
     * @var string
     */
    private $_privada = null;
    /**
     * @access protected
     * @var int
     */
    protected $protegida = 0;

    /**
     * @var string
     */
    public $publica  = null;
    /**
     * @static
     * @var string
     */
    public static $publicaEstatica = null;
    // Qualquer conteúdo da classe
    // precisa ser identado com [Tab|4 espaços].
}
```

## Variáveis de Classes (Propriedades)

- Nomear em lowerCamelCase
- Devem ser declaradas no topo da classe, antes de qualquer declaração de métodos.
- Não utilizar o identificador: var (php4)
- Sempre declarar sua visibilidade: private, protected ou public.
- Preferencialmente não utilizar declaração de variáveis de classes como public, para incentivar o uso de (set/get)

- Funções e/ou Métodos
  - Nomear em lowerCamelCase.
  - As chaves "{" e "}" virão na linha abaixo do nome da Função/Método.
  - Toda Função/Método deve ter um bloco de documentação em conformidade com o Padrão do PHPDocumentor.
  - Qualquer código dentro da Função/Método precisa ser identado com quatro espaços.
  - Sempre declarar a visibilidade: private, protect ou public.

```php
/**
 * Bloco de Documentação
 */
class Exemplo
{
    protected $name = null;
    /**
     * Seta o um valor à propriedade Name.
     * @access public
     * @param string $pValue Nome Completo
     */
    public function setName( $pValue )
    {
        // Qualquer conteúdo
        // precisa ser identado com [Tab|4 espaços].
    }
    /**
     * Retorna o valor da propriedade Name.
     * @access public
     * @return string
     */
    public function getName()
    {
        // Qualquer conteúdo
        // precisa ser identado com [Tab|4 espaços].
    }
}
```

## Estruturas de Controle

### if / else / elseif

- Usar espaço simples depois do "(" e antes do ")" da condiçao no IF e no ELSEIF
- A chave "{" virá na mesma linha da expressão, a chave "}" virá na linha abaixo da última linha de conteúdo.
- Qualquer código entre as chaves "{" e "}" precisa ser identado com quatro espaços.

```
if ( $intValor == 1 ) {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
} elseif ( ( $intValor == 2 ) or ( $intValor == 3 ) ) {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
} else {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
}
```

### while/for/foreach

- Usar espaço simples depois do "(" e antes do ")" da expressão.
- A chave "{" virá na mesma linha da expressão, a chave "}" virá na linha abaixo da última linha de conteúdo.
- Qualquer código entre as chaves "{" e "}" precisa ser identado com quatro espaços.

```
while ( !feof( $hFile ) ) {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
}
```

```
for ( $intCount = 0; $intCount <= 10; $intCount++ ) {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
}
```

```
foreach ( $arrList as $mixKey => $mixValue ) {
    // Qualquer conteúdo
    // precisa ser identado com [Tab|4 espaços].
}
```

- switch/case
  - Usar espaço simples depois do "(" e antes do ")"
  - A chave "{" virá na mesma linha da Expressão, a chave "}" virá na linha abaixo da última linha de conteúdo
  - Qualquer código entre as chaves "{" e "}" precisa ser identado com quatro espaços.
  - Qualquer código dentro de: *case* e *default*, precisa ser identado com quatro espaços, inclusive a palavra reservada: *break.*

```
switch ( $intNivel ) {
    case 1:
        // Qualquer conteúdo
        // precisa ser identado com [Tab|4 espaços].
        break;
    case 2:
        // Qualquer conteúdo
        // precisa ser identado com [Tab|4 espaços].
        break;
    default:
        // Qualquer conteúdo
        // precisa ser identado com [Tab|4 espaços].
}
```

# Licença

- Este conteúdo está sob a licença ***Creative Commons: BY-NC-SA***.
  - Sendo permitido:
    - copiar, distribuir, exibir e executar a obra.
    - criar obras derivadas.
  - Sob as seguintes condições:
    - **Atribuição**. Você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante.
    - **Uso Não-Comercial**. Você não pode utilizar esta obra com finalidades comerciais.
    - **Compartilhamento pela mesma Licença**. Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.
    - Para cada novo uso ou distribuição, você deve deixar claro para outros os termos da licença desta obra.
    - Qualquer uma destas condições pode ser renunciada, desde que Você obtenha permissão do autor.
    - Nada nesta licença prejudica ou restringe os direitos morais do autor.
  - Anexo:
    - Ao adotar este padrão em alguma instituição informar ao autor através do e-mail: walkeralencar@gmail.com ou contato@walkeralencar.com, para ser referenciado no site, isso engrandeçe o trabalho já realizado e incentiva continuidade de outros mais.

# Referência


- Zend Framework: Coding Standards for PHP
- PEAR Coding Standards
- Drupal Coding Standards
- Coding Standards for CakePHP
- eZ Coding Standards for PHP
- Wikipedia: CamelCase