

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Técnicas de Programação

Profº: Mauricio Serrano

Alunos: Ana Carolina Lopes da Silva Mat: 09/0078659

Cristiano Costa de Sousa Mat: 10/44320

Fabrício Soares Coelho Mat: 11/0116534

**Folha de estilo PHP**

# Visão Geral

* Códigos DEVEM seguir [PSR-1](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md).
* Códigos DEVEM utilizar tabs, equivalente à 4 espaços para indentação.
* As linhas DEVEM ter 80 caracteres ou menos.
* DEVE existir uma linha em branco após da declaração do namespace e DEVE existir uma linha em branco após o bloco de declarações de use.
* Chaves de abertura para classes DEVEM ser colocadas na linha seguinte e chaves de fechamento DEVEM ser colocadas na linha após o corpo da classe.
* Chaves de abertura para métodos DEVEM ser colocadas na linha seguinte e chaves de fechamento DEVEM ser colocadas na linha após o corpo do método.
* Visibilidade DEVE ser declarada em todas as propriedades e métodos; abstract e final DEVEM ser declarados antes da visibilidade; static DEVE ser declarado depois da visibilidade.
* Palavras-chave de estruturas de controle DEVEM ter um espaço depois delas; chamadas de métodos e funções NÃO DEVEM.
* Chaves de abertura para estruturas de controle DEVEM ser colocadas na mesma linha e chaves de fechamento DEVEM ser colocadas na linha após o corpo da estrutura de controle.
* Parênteses de abertura para estruturas de controle NÃO DEVEM ter um espaço depois delas e parênteses de fechamento para estruturas de controle NÃO DEVEM ter um espaço antes.

Exemplo:

**<?php**

**namespace** Vendor\Package;

**use** FooInterface;

**use** BarClass **as** Bar;

**use** OtherVendor\OtherPackage\BazClass;

**class** **Foo** **extends** Bar **implements** FooInterface

{

**public** **function** **sampleFunction**($a, $b **=** **null**)

{

**if** ($a **===** $b) {

bar();

} **elseif** ($a **>** $b) {

$foo**->**bar($arg1);

} **else** {

BazClass**::**bar($arg2, $arg3);

}

}

**final** **public** **static** **function** **bar**()

{

*// corpo do método*

}

}

## Arquivos

Todos os arquivos PHP DEVEM utilizar o padrão Unix LF (linefeed) de terminação de linhas.

Todos os arquivos PHP DEVEM terminar com uma única linha em branco.

A tag de fechamento ?> DEVE ser omitida em arquivos contendo somente PHP.

## Indentação

A indentação DEVE ser feita utilizando tab (equivalente à 4 espaços).

## Linha

A linha deverá ter um máximo de 80 caracteres.

Linhas NÃO DEVERIAM ser mais longas que 80 caracteres; Linhas mais longas que isso DEVERIAM ser divididas em múltiplas linhas subsequentes de não mais que 80 caracteres cada.

NÃO DEVE haver espaços em branco ao final de linhas que não estão em branco.

Linhas em branco PODEM ser adicionadas para melhorar a legibilidade e para indicar blocos relacionados de código.

NÃO DEVE haver mais do que uma declaração por linha.

## Palavras-chave e True/False/Null

[Palavras-chave](http://php.net/manual/en/reserved.keywords.php) PHP DEVEM ser em letras minúsculas.

As constantes PHP true, false e null DEVEM ser em letras minúsculas.

## EOL Convention (Fim da linha)

Os finais de linha DEVEM seguir o padrão UNIX, elas devem terminar com um único linefeed(LF - /n).

## Declarações Namespace e Use

Quando presente, DEVE haver uma linha em branco após a declaração do namespace.

Quando presente, todas as declarações use DEVEM ser colocadas após a declaração donamespace.

DEVE haver somente uma palavra-chave use por declaração.

DEVE haver uma linha em branco depois do bloco de use.

Exemplo:

**<?php**

**namespace** Vendor\Package;

**use** FooClass;

**use** BarClass **as** Bar;

**use** OtherVendor\OtherPackage\BazClass;

*// ... mais código PHP ...*

# Classes, Propriedades e Métodos

O termo classe se refere a todas as classes, interfaces e traits.

## Extends e Implements

As palavras-chave extends e implements DEVEM ser declaradas na mesma linha que o nome da classe.

A chave de abertura para a classe DEVE ser colocada em sua própria linha; a chave de fechamento DEVE ser coloca na linha após o corpo da classe.

**<?php**

**namespace** Vendor\Package;

**use** FooClass;

**use** BarClass **as** Bar;

**use** OtherVendor\OtherPackage\BazClass;

**class** **ClassName** **extends** ParentClass **implements** \ArrayAccess, \Countable

{

*// constantes, propriedades, métodos*

}

Listas de implements PODEM ser divididas em múltiplas linhas, onde cada linha subsequente é indentada uma vez. Quando fazendo isto, o primeiro item da lista DEVE ser colocado na linha seguinte e DEVE haver somente uma interface por linha.

**<?php**

**namespace** Vendor\Package;

**use** FooClass;

**use** BarClass **as** Bar;

**use** OtherVendor\OtherPackage\BazClass;

**class** **ClassName** **extends** ParentClass **implements**

\ArrayAccess,

\Countable,

\Serializable

{

*// constantes, propriedades, métodos*

}

## Métodos

Visibilidades DEVEM ser declaradas em todos os métodos.

Nomes de métodos NÃO DEVERIAM ser prefixadas com \_ para indicar visibilidades protected ouprivate.

Assinaturas de métodos NÃO DEVEM ser declaradas com um espaço após o nome do método. A chave de abertura DEVE ser colocada em sua própria linha e a chave de fechamento DEVE ser colocada na linha após o corpo do método. NÃO DEVE haver um espaço depois do parenteses de abertura e NÃO DEVE haver um espaço antes do parenteses de fechamento.

Uma declaração de método se parece com o seguinte. Note o posicionamento dos parenteses, virgulas, espaços e chaves:

**<?php**

**namespace** Vendor\Package;

**class** **ClassName**

{

**public** **function** **fooBarBaz**($arg1, **&**$arg2, $arg3 **=** [])

{

*// corpo do método*

}

}

## Argumentos de Métodos

Na lista de argumentos, NÃO DEVE haver um espaço antes de cada vírgula e DEVE haver um espaço após cada vírgula.

Argumentos de métodos com valores default DEVEM ser colocados ao fim da lista de argumentos.

**<?php**

**namespace** Vendor\Package;

**class** **ClassName**

{

**public** **function** **foo**($arg1, **&**$arg2, $arg3 **=** [])

{

*// corpo do método*

}

}

Lista de argumentos PODEM ser divididas entre múltiplas linhas, onde cada linha subsequente é indentada uma vez. Quando fazendo isto, o primeiro item da lista DEVE estar na linha seguinte e DEVE haver somente um argumento por linha.

Quando a lista de argumento é dividida em multiplas linhas, o parenteses de fechamento e a chave abertura DEVEM ser colocadas juntas em sua própria linha com um espaço entre elas.

**<?php**

**namespace** Vendor\Package;

**class** **ClassName**

{

**public** **function** **aVeryLongMethodName**(

ClassTypeHint $arg1,

**&**$arg2,

**array** $arg3 **=** []

) {

*// corpo do método*

}

}

# Estruturas de Controle

As regras gerais de estilo para estruturas de controle são as seguintes:

* DEVE haver um espaço após a palavra-chave da estrutura de controle
* NÃO DEVE haver um espaço depois do parênteses de abertura
* NÃO DEVE haver um espaço antes do parênteses de fechamento
* DEVE haver um espaço entre o parênteses de fechamento e a chave de abertura.
* O corpo da estrutura DEVE ser indentada uma vez
* A chave de fechamento DEVE ser colocada na linha após o corpo da estrutura

O corpo de cada estrutura DEVE ser envolta por chaves. Isso padroniza como as estruturas se parecem e reduz a possibilidade de introduzir erros à medida que novas linhas são adicionadas ao corpo da estrutura.

## if, elseif, else

Uma estrutura if se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves; e que else e elseif estão na mesma linha que a chave de fechamento do corpo da estrutura anterior.

**<?php**

**if** ($expr1) {

*// corpo do if*

} **elseif** ($expr2) {

*// corpo do elseif*

} **else** {

*// corpo do else*

}

A palavra-chave elseif DEVERIA ser utilizada ao invés de else if para que todas as palavras-chave de controle se pareçam com uma só palavra.

## switch, case

Uma estrutura switch se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves. A declaração case DEVE ser identada uma vez do switch e a palavra-chave case (ou qualquer outra palavra-chave de terminação) DEVE ser indentada no mesmo nível que o corpo docase. DEVE haver um comentário como //sem break quando a passagem próximo case é intencional em um corpo de case que não está vazio.

**<?php**

**switch** ($expr) {

**case** 0**:**

**echo** 'Primeiro case, com um break';

**break**;

**case** 1**:**

**echo** 'Segundo case, passando para o próximo case';

*// sem break*

**case** 2**:**

**case** 3**:**

**case** 4**:**

**echo** 'Terceiro case, return ao invés de break';

**return**;

**default:**

**echo** 'Default case';

**break**;

}

## while, do while

Uma estrutura while se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves.

**<?php**

**while** ($expr) {

*// corpo da estrutura*

}

Similarmente, uma estrutura do while se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves.

**<?php**

**do** {

*// corpo da estrutura*

} **while** ($expr);

## for

Uma estrutura for se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves.

**<?php**

**for** ($i **=** 0; $i **<** 10; $i**++**) {

*// corpo do for*

}

## foreach

Uma estrutura foreach se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves.

**<?php**

**foreach** ($iterable **as** $key **=>** $value) {

*// corpo do foreach*

}

## try, catch

Uma estrutura try catch se parece com o seguinte. Note o posicionamento dos parenteses, espaços e chaves.

**<?php**

**try** {

*// corpo do try*

} **catch** (FirstExceptionType $e) {

*// corpo do catch*

} **catch** (OtherExceptionType $e) {

*// corpo do catch*

}

\*Padrão de Código utilizado PSR-1, material disponível em: http://www.php-fig.org/psr/psr-2/pt-br/

**Folha de estilo JavaScript**

## Costrutores

Adotamos a primeira letra sempre em maiusculo:

var joao = new Pessoa();

## **Funções e métodos**

Adotamos a primeira letra em minúsculo e a primeira letra das palavras seguintes em maiúsculo, sem espaço ou traços entre palavras. Este formato é comumente chamado de "camelo" ou [Camel Case](http://en.wikipedia.org/wiki/CamelCase):

function meuMetodoTeste() {...}  
function minhaFuncaoTeste() {...}

## Variáveis

Adotamos todas as letras em minúsculo, porém as palavras são separadas por um underscore.

var minha\_variavel\_numero = 1;

## Espaços ao redor de Operadores

Sempre colocar espaços aoredore de operadores (= + / \* ), e depos de vígulas:

var x = y + z;  
var values = ["Volvo", "Saab", "Fiat"];

## Indentação do Código

Sempre usar 4 espaços para indentaçao

function toCelsius(fahrenheit) {  
    return (5/9) \* (fahrenheit-32);  
}

## Regras de Declaração

* Sempre finalizar uma declaração simples com ponto e vírgula.

var values = ["Volvo", "Saab", "Fiat"];

var person = {  
    firstName: "John",  
    lastName: "Doe",  
    age: 50,  
    eyeColor: "blue"  
};

* Abrir chave no final da primeira linha
* Dar um espaço antes da abertura de chave
* Fechar a chave em uma nova linha, sem espaços
* Não finalizar a declaração composta com ponto e vírgula

function toCelsius(fahrenheit) {  
    return (5/9) \* (fahrenheit-32);  
}

for (i = 0; i < 5; i++) {  
    x += i;  
}

if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}

## Objetos

Regras Gerais para definir objetos:

* Colocar a abertura de chave na mesma linha do nome do objeto
* Colocar dois pontos mais um espaço entre cada propriedade e o seu repectivo valor
* Usar aspas ao redor de valores String, e não usar ao redor de valores numéricos
* Não adicionar vírgula depois do valor da última propriedade
* Sempre terminar a definição de um objeto com ponto e vírgul

var person = {  
    firstName: "John",  
    lastName: "Doe",  
    age: 50,  
    eyeColor: "blue"  
};

* Pequenos objetos podem ser escritos de maneira comprimida em apenas uma linha

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

## Tamanho da linha

Para facilitar a leitura, evite linhas com mais de 80 caracteres.

Se uma declaração em JavaScript não couber em uma linha, o melhor lugar para quebrar a linha será após um operador ou uma vírgula.

document.getElementById("demo").innerHTML =  
    "Hello Dolly.";

## Convenção de nomeclaturas

Sempre use a mesma convenção de nomeclaturas em todo código:

* Nomes de variáveis e funções escritas em camelCase
* Variaveis Globais escritas em MAIÚSCULO
* Constantes escritas em MAIÚSCULO