PSR-2: Guia de Estilo de Codificação (*Coding Style Guide*)

Este guia é uma extensão do PSR-1 - Padrão Básico de Codificação.

A intenção deste guia é reduzir a discrepância quando códigos de diferentes autores são reaproveitados. Isto é feito utilizando uma série de regras compartilhadas e expectativas sobre como formatar código PHP.

Os tipos das regras são compartilhadas pelos vários membros de projeto. Quando diferentes autores colaboram em múltiplos projetos, ter um guia para ser seguido, auxília na implementação destes projetos. Além disso, o objetivo deste guia não são as regras em si, mas, o compartilhamento destas regras.

As palavras chaves “É OBRIGATÓRIO” (*MUST*) , “NÃO É PERMITIDO” (*MUST NOT*), “É REQUERIDO” (*REQUIRED*), “DEVEM” (*SHALL*), “NÃO DEVEM” (*SHALL NOT*), “DEVERIA” (*SHOULD*), “NÃO DEVERIA” (*SHOULD NOT*), “É RECOMENDADO” (*RECOMMENDED*), “PODE” (*MAY*), e "OPCIONAL” (*OPTIONAL*) nesse documento devem ser interpretadas como no RFC 2119.

1. Visão Geral

- Código “É OBRIGATÓRIO” seguir o “*coding style guide*” (guia

de estilo de codificalçao) PSR [PSR-1].

- Código “É OBRIGATÓRIO” utiliza 4 espaços de indentação, sem *tabs*.
- “NÃO É PERMITIDO” ter um limite rígido de comprimento/tamanho de linha; o limite “É OBRIGATÓRIO” ser de 120 caracteres; linhas “DEVERIA” ser de 80 caracteres ou menos.
- “É OBRIGATÓRIO” ter uma linha em branco depois de uma declaração de *namespace*, e “É OBRIGATÓRIO” ter uma linha em branco depois de um bloco de declaração *use*.
- Abre chave ({) ({ } para classes “É OBRIGATÓRIO” ser na próxima linha, e fecha chave (}) “É OBRIGATÓRIO” ser na próxima linha depois do *body* (corpo de código).
- Abre chave ({) para métodos “É OBRIGATÓRIO” ser na próxima linha, e fecha chave (}) “É OBRIGATÓRIO” ser na próxima linha depois do *body* (corpo de código).
- *Visibility* (Visiabilidade) “É OBRIGATÓRIO” ser declaradas com todas as propriedades e métodos; *abstract* e final “É OBRIGATÓRIO” serem declaradas antes da *visibility* (visiabilidade); *static* “É OBRIGATÓRIO” ser declarado depois da *visibility* (visiabilidade).
- Palvras chaves de estrutura de controle “É OBRIGATÓRIO” ter um espaço depois de cada declaração; “NÃO É PERMITIDO” espaço nas chamadas de métodos e funções.
- Abre chave ({) para estruturas de controle “É OBRIGATÓRIO” serem na mesma linha, e fecha chave (}) deve ir na próxima linha depois do *body* (corpo de código).
- Abre parênteses (( ) para estruturas de controle “NÃO É

PERMITIDO” ter um espaço depois de cada declaração, e fecha parênteses para estruturas de controle “NÃO É PERMITIDO” terem um espaço antes de cada declaração.

1.1. Exemplo

Este exemplo exemplefica algumas das regras citadas acima:

```
<?php
namespace Vendor\Package;

use FooInterface;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class Foo extends Bar implements FooInterface
{
public function sampleMethod($a, $b = null)
{
if ($a === $b) {
bar();
} elseif ($a > $b) {
$foo->bar($arg1);
} else {
BazClass::bar($arg2, $arg3);
}
}
```

```
final public static function bar()
{
    // método body
}
```

}

2. *General* (Genérico)

2.1. Padrão Básico de Codificação

Código “É OBRIGATÓRIO” obedecer todas as regras determinadas em PSR-1.

2.2. Arquivos

Todos arquivos PHP “É OBRIGATÓRIO” utilizar o Unix LF (*linefeed*) como fim de linha.

Todos arquivos PHP “É OBRIGATÓRIO” terminar com uma única linha em branco.

O fechamento da *tag* `?>` “É OBRIGATÓRIO” ser omitida em arquivos que contém unicamente PHP.

2.3. Linhas

“NÃO É PERMITIDO” ter um limite rígido de tamanho/comprimento de

linha.

O limite básico de tamanho/comprimento de linha “É OBRIGATÓRIO” ser de 120 caracteres; verificadores automático de estilo (*automated style checkers*) “É OBRIGATÓRIO” alertar (*warning*) mas “NÃO É PERMITIDO” dar erro (*error*) quando utilizado este limite básico.

Linhas “NÃO DEVERIA” ser maiores do que 80 caracteres; linhas maiores que 80 caracteres “DEVERIA” ser quebradas/separadas em múltiplas linhas subsequentes menores que 80 caracteres.

“NÃO É PERMITIDO” ter espaço em branco em fim de linhas que não são em branco.

Linhas em branco “PODE” ser adiciondas para melhorar a legibilidade e para indicar blocos de código.

“NÃO É PERMITIDO” ter mais que uma declaração por linha.

2.4. Indentação

Código “É OBRIGATÓRIO” usar 4 espaços de indentação, e “NÃO É PERMITIDO” usar *tabs* para indentação.

Utilizar somentes espaços, e não misturar espaços com *tabs*, contribui para evitar problemas de *diffs*, *patches*, *history* (histórico), e anotações. O uso de espaços também facilita a utilização de sub-indentações para alinhar linhas internas do bloco de código.

2.5. Palavras Chaves e *True* (Verdadeiro)/*False* (Falso)/*Null* (Vazio)

Palavras chaves PHP “É OBRIGATÓRIO” ser em caixa baixa (letra minúscula).

Constants true, *false*, and *null* (constantes/verdadeiro/falso/vazio) “É OBRIGATÓRIO” serem em caixa baixa (letra minúscula).

3. *Namespace* e Declarações

“É OBRIGATÓRIO” ter uma linha em branco depois de uma declaração de *namespace*.

“É OBRIGATÓRIO” que todas as declarações que não sejam de *namespace* sejam declaradas depois da declaração de *namespace*.

“É OBRIGATÓRIO” que utilize somente uma palavra chave para cada declaração.

“É OBRIGATÓRIO” ter apenas uma linha em branca depois de um bloco *use*.

Por exemplo:

```
<?php
```

namespace Vendor\Package;

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

// ... Código PHP adicional ...

4. Classes, Propriedades, e Métodos

O termo “classe” refere-se a todas as classes, interfaces e *traits*.

4.1. Extensões e Implementação

As extensões e palavras chaves de implementação “É OBRIGATÓRIO” serem declaradas na mesma linha e com o mesmo nome de classe.

Abrir chave para classe “É OBRIGATÓRIO” ser na mesma linha; fechar chave para classe “É OBRIGATÓRIO” ser na próxima linha depois do *body*.

```
<?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 implemetações “PODE” ser quebrada em múltiplas linhas, sendo que cada linha é indentanda somente uma vez. O primeiro item da lista “É OBRIGATÓRIO” ser na próxima linha, e “É OBRIGATÓRIO” ter uma única 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
}
```

4.2. Propriedades

Visibility (visiabilidade) “É OBRIGATÓRIO” ser declarada com todas as propriedades.

A palavra chave *var* “NÃO É PERMITIDO” ser usada para declarar uma

propriedade.

“NÃO É PERMITIDO” que tenha mais de uma propriedade por declaração.

Nomes de propriedades “NÃO DEVERIA” ser um prefixo com um único subtraço (_) para indicar *protected* (protegido) ou *private* (privado) *visibility* (visiabilidade).

Uma declaração de propriedade é feita como no exemplo abaixo:

```
<?php
namespace Vendor\Package;

class ClassName
{
public $foo = null;
}
```

4.3. Métodos

Visibility (Visiabilidade) “É OBRIGATÓRIO” ser declarada em todos os métodos.

Nomes de métodos “NÃO DEVERIA” ser um prefixo com um único subtraço (_) para indicar *protected* (protegido) ou *private* (privado) *visibility* (visiabilidade).

Nomes de métodos “NÃO É PERMITIDO” ser declarados com uma espaço após um nome de método. Abre chave ({) “É OBRIGATÓRIO” irem na mesma linha da declaração do método, e fecha chave (}) “É OBRIGATÓRIO” ser na linha após o *body*. “NÃO É PERMITIDO” ter um espaço em branco após abrir parênteses, e “NÃO É PERMITIDO” ter um espaço em branco antes de fechar parênteses.

Segue como seria uma declaração de método. Note a posição dos parênteses, vírgulas, espaços e chaves ({ }):

```
<?php
namespace Vendor\Package;

class ClassName
{
public function fooBarBaz($arg1, &$arg2, $arg3 = [])
{
// método body
}
}
```

4.4. Argumentos dos Métodos

Uma lista de argumentos “NÃO É PERMITIDO” ter um espaço antes de cada vírgula (,), e “É OBRIGATÓRIO” ter um espaço após cada vírgula (,).

Argumentos de métodos com valores *default* (padrões) “É

OBRIGATÓRIO” estarem no final da lista de argumentos.

```
<?php
namespace Vendor\Package;

class ClassName
{
public function foo($arg1, &$arg2, $arg3 = [])
{
// método body
}
}
```

Listas de argumentos “PODE” ser quebradas em múltiplas linhas, em que cada linha é indentada uma única vez. O primeiro item da lista “É OBRIGATÓRIO” estar na próxima linha, e “É OBRIGATÓRIO” ter um único argumento por linha.

Quando a lista de argumentos é quebrada em múltiplas linhas, fecha ( )) parênteses e abre chave ({) ({) “É OBRIGATÓRIO” estarem juntos na mesma linha com um espaço entre eles.

```
<?php
namespace Vendor\Package;

class ClassName
{
public function aVeryLongMethodName(
ClassTypeHint $arg1,
```

&$arg2,

array $arg3 = []

) {

// método *body*

}

}

4.5. *abstract* (resumo), *final*, e *static* (estático)

As declarações *abstract* (resumo) e *final* "É OBRIGATÓRIO" preceder a declaração de *visibility* (visiabilidade).

A declaração *static* (estático) "É OBRIGATÓRIO" ser feita depois da declaração de *visibility* (visiabilidade).

`<?php`

namespace Vendor\Package;

abstract class ClassName

{

protected static $foo;

```
abstract protected function zim();

final public static function bar()
{
    // método _body_
```

```
}
```

}

4.6. Métodos e Chamadas de Funções

Quando for criar um método ou uma chamada de função, “NÃO É PERMITIDO” ter um espaço entre o método ou o nome da função e no abre parênteses (), “NÃO É PERMITIDO” ter um espaço após abre parênteses (), e “NÃO É PERMITIDO” ter um espaço antes de fechar parênteses ()). Na lista de argumentos “NÃO É PERMITIDO” ter um espaço antes de cada vírgula, e “É OBRIGATÓRIO” ter um espaço após cada vírgula.

```
<?php
bar();
$foo->bar($arg1);
Foo::bar($arg2, $arg3);
```

Lista de argumentos “PODE” ser quebrada em múltiplas linhas, cada sublinha é indentada uma únicaz vez. O primeiro item da lista “É OBRIGATÓRIO” estar na próxima linha, e “É OBRIGATÓRIO” ter somente um argumento por linha.

```
<?php
$foo->bar(
$longArgument,
$longerArgument,
```

$muchLongerArgument
);

5. Estruturas de Controle

As regras genéricas para estruturas de conrole são as seguintes:

- “É OBRIGATÓRIO” ter um espaço após uma palavra chave de estrutura de controle.
- “NÃO É PERMITIDO” ter um espaço após abre parênteses (( ).
- “NÃO É PERMITIDO” ter um espaço antes fecha parênteses ( )).
- “É OBRIGATÓRIO” ter uma espaço entre fecha parênteses ( )) e abre chave ({).
- A estrutura de *body* “É OBRIGATÓRIO” ser indentada uma única vez.
- Fecha chave (}) “É OBRIGATÓRIO” estar na linha após o *body*.

O *body* de cada estrutura “É OBRIGATÓRIO” estar entre chaves ({ e }). Este padrão de estrutura reduz os erros quando novas linhas são adicionadas ao *body*.

5.1. *if* , *elseif* , *else*

Uma estrutura de *if* é exemplificada a seguir. Note a posição dos parênteses, espaços e chaves ({ e }); e que *else* e *elseif* estão na mesma linha.

```
<?php
if ($expr1) {
// if body
} elseif ($expr2) {
// elseif body
} else {
// else body;
}
```

A palavra chave *elseif* "PODERIA" ser usada ao invés de *else if*, assim todas as palavras chaves de controle seriam a mesma palavra.

5.2. *switch*, *case*

Uma estrutura de *switch* é exemplificada a seguir. Note a posição dos parênteses, espaços e chaves. "É OBRIGATÓRIO" a estrutura ser indentada uma vez para o *switch* e outra para o *break* (ou outra palavra chave de finalização) "É OBRIGATÓRIO" ser indentada no mesmo level do *body*. "É OBRIGATÓRIO" ter um comentário do tipo *// no break* quando uma finalização é intencional em um *non-empty* (não vazio) *body*.

```
<?php
switch ($expr) {
case 0:
echo 'First case, with a break';
break;
case 1:
```

```
echo ‘Second case, which falls through’;
// no break
case 2:
case 3:
case 4:
echo ‘Third case, return instead of break’;
return;
default:
echo ‘Default case’;
break;
}
```

5.3. *while , do while*

Uma estrutura de *while* é exemplificada a seguir. Note a posição dos parênteses, espaõs e chaves ({ e }).

```
<?php
while ($expr) {
// estrutura do body
}
```

Uma estrutura de *do while* é exemplificada a seguir. Note a posição dos parênteses, espaõs e chaves ({ e }).

```
<?php
do {
// estrutura do body
```

} while ($expr);

5.4. *for*

Uma estrutura de *for* é exemplificada a seguir. Note a posição dos parênteses, espaõs e chaves ({ e }).

```
<?php
for ($i = 0; $i < 10; $i++) {
// for body
}
```

5.5. *foreach*

Uma estrutura de *foreach* é exemplificada a seguir. Note a posição dos parênteses, espaõs e chaves ({ e }).

```
<?php
foreach ($iterable as $key => $value) {
// foreach body
}
```

5.6. *try, catch*

Um bloco de *try catch* é exemplifica a seguir. Note a posição dos parênteses, espaõs e chaves ({ e }).

```
<?php
try {
```

```
// try body
} catch (FirstExceptionType $e) {
// catch body
} catch (OtherExceptionType $e) {
// catch body
}
```

6. *Closures* (Fechamentos)

Closures “É OBRIGATÓRIO” ser declara com espaço após uma palavra chave de função, e um espaço antes e após o uso de uma palavra chave.

Abre chave ({) “É OBRIGATÓRIO” ser na mesma linha, e o fecha chave (}) “É OBRIGATÓRIO” ir na linha após o *body*.

“NÃO É PERMITIDO” ter um espaço após abre parênteses (() na lista de argumentos ou na lista de variáveis, e “NÃO É PERMITIDO” ter um espaço antes de fechar parênteses ()) na lista de argumentos ou na lista de variáveis.

Na lista de argumentos e na lista de variáveis, “NÃO É PERMITIDO” ter um espaço antes de cada vírgula, e “É OBRIGATÓRIO” ter um espaço após cada vírgula.

Argumentos *Closure* com valores padrões “É OBRIGATÓRIO” irem no final da lista de argumentos.

Uma declaração de *closure* é exemplificada a seguir. Note a posicação dos parênteses, vírgulas, espaçõs e chaves.

```
<?php
$closureWithArgs = function ($arg1, $arg2) {
// body
};

$closureWithArgsAndVars = function ($arg1, $arg2) use ($var1, $var2)
{
// body
};
```

Lista de argumentos e lista de variáveis “PODE” ser quebradas em múltiplas linhas, cada sublinha é indentada uma única vez. O primeiro item da lista “É OBRIGATÓRIO” estar na próxima linha, e “É OBRIGATÓRIO” ter somente um argumento ou variável em cada linha.

Quando o fim da lista é quebrada em múltiplas linhas, o fecha parênteses ()) e abre chaves ({) “É OBRIGATÓRIO” estarem juntos na mesma linha com um espaço entre eles.

Segue um exemplo de *closures* com/sem lista de argumentos e variáveis quebradas em múltiplas linhas.

```
<?php
$longArgs_noVars = function (
```

```
$longArgument,
$longerArgument,
$muchLongerArgument
) {
// body
};

$noArgs_longVars = function () use (
$longVar1,
$longerVar2,
$muchLongerVar3
) {
// body
};

$longArgs_longVars = function (
$longArgument,
$longerArgument,
$muchLongerArgument
) use (
$longVar1,
$longerVar2,
$muchLongerVar3
) {
// body
};

$longArgs_shortVars = function (
```

```
$longArgument,
$longerArgument,
$muchLongerArgument
) use ($var1) {
// body
};

$shortArgs_longVars = function ($arg) use (
$longVar1,
$longerVar2,
$muchLongerVar3
) {
// body
};
```

Note que a formatação das regras também se aplicam quando o *closure* é usado diretamente em uma função ou numa chamada de método como um argumento.

```
<?php
$foo->bar(
$arg1,
function ($arg2) use ($var1) {
// body
},
$arg3
);
```

7. Conclusão

Há vários elementos de estilo e de prática itencionalmente omitidas neste guia. Segue as que estão incluídas (mas, não são as únicas):

- Declaração de variáveis e constantes globais.
- Declaração de funções.
- Operadores e *assignment* (atribuição).
- Indentação de linhas.
- Comentários e documentação em blocos de código.
- Prefixos e sufixos de nomes de classes.
- Boas práticas.
- Recomendações futuras deste guia “PODE” ser revisadas, para que aja uma melhoria dos elementos de estilos e da prática.

8. Apêndice

Para escrever este guia de estilos, o grupo fez uma pesquisa com os membros de projetos que possuem práticas comuns de codificação.

8.1. Dados da Pesquisa

url,http://www.horde.org/apps/horde/docs/CODING_STANDARDS,http://pe
standards.style,http://framework.zend.com/manual/en/coding-
standard.html,http://symfony.com/doc/2.0/contributing/code/standards.htr
standards.html,https://github.com/ezsystems/ezp-
next/wiki/codingstandards,http://book.cakephp.org/2.0/en/contributing/ca
coding-
conventions.html,https://github.com/UnionOfRAD/lithium/wiki/Spec%3A-

Coding,http://drupal.org/coding-standards,http://code.google.com/p/sabredav/,http://area51.phpbb.com/do guidelines.html,https://docs.google.com/a/zikula.org/document/edit?authkey=CPCU0Us&hgd=1&id=1fcqb93Sn-hR9c0mkN6m_tyWnmEvoswKBtSc0tKkZmJA,http://www.chisimba.com,n/a info/blob/master/coding-standards-sample.php,n/a,Object Calisthenics for PHP,http://doc.nette.org/en/coding-standard,http://flow3.typo3.org,https://github.com/propelorm/Propel2/wik Standards,http://developer.joomla.org/coding-standards.html

voting,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,yes,no,no,nc

indent_type,4,4,4,4,4,tab,4,tab,tab,2,4,tab,4,4,4,4,4,4,tab,tab,4,tab

line_length_limit_soft,75,75,75,75,no,85,120,120,80,80,80,no,100,80,80,?,

line_length_limit_hard,85,85,85,85,no,no,no,no,100,?,no,no,no,100,100,?,1

class_names,studly,studly,studly,studly,studly,studly,studly,studly,studly,stu

class_brace_line,next,next,next,next,next,same,next,same,same,same,sam

constant_names,upper,upper,upper,upper,upper,upper,upper,upper,upper,upper,u

true_false_null,lower,lower,lower,lower,lower,lower,lower,lower,lower,uppe

method_names,camel,camel,camel,camel,camel,camel,camel,camel,camel,camel

method_brace_line,next,next,next,next,next,same,next,same,same,same,s

control_brace_line,same,same,same,same,same,same,next,same,same,san

control_space_after,yes,yes,yes,yes,yes,no,yes,yes,yes,yes,no,yes,yes,yes,y

always_use_control_braces,yes,yes,yes,yes,yes,yes,no,yes,yes,yes,no,yes,y

else_elseif_line,same,same,same,same,same,same,next,same,same,next,sa

case_break_indent_from_switch,0/1,0/1,0/1,1/2,1/2,1/2,1/2,1/1,1/1,1/2,1/2,

function_space_after,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,no,

closing_php_tag_required,no,no,no,no,no,no,no,no,yes,no,no,no,no,yes,no,

line_endings,LF,LF,LF,LF,LF,LF,LF,LF,?,LF,?,LF,LF,LF,LF,?,LF,?,LF,LF,LF

static_or_visibility_first,static,?,static,either,either,either,visibility,visibility

control_space_parens,no,no,no,no,no,no,yes,no,no,no,no,no,no,yes,?,no,nc

blank_line_after_php,no,no,no,no,yes,no,no,no,no,yes,yes,no,no,yes,?,yes,y

class_method_control_brace,next/next/same,next/next/same,next/next/sam

8.2. Legenda da Pesquisa

indent_type: tipo de indentação. *tab* = “Utilizar tab”, 2 ou 4 = “número de espaços”

line_length_limit_soft: O comprimento limite "*soft*" da linha, em caracteres. ? = não há limite.

line_length_limit_hard: O comprimento limite "*hard*" da linha, em caracteres. ? = não há limite.

class_names: Como são chamadas os nomes das classes? *lower* = *lowercase* , *lower_under* = *lowercase* (com subtraços como separadores), *studly* = *StudlyCase*.

class_brace_line: O abre chave ({) para uma classe está na mesma linha da palavra chave *class*, ou está na próxima linha?

constant_names: Como são os nomes das constantes? *upper* = *Uppercase* (com subtraços como separadores).

true_false_null: As palavras chaves *true*, *false*, e *null* são todas escritas como em *lower case*, ou em *upper case*?

method_names: Como são os nomes dos métodos? *camel* = *camelCase*, *lower_under* = *lowercase* (com subtraços como separadores).

method_brace_line: O abre chave ({) para um método está na mesma linha do nome do método, ou está na próxima linha?

control_brace_line: O abre chave ({) para estrutura de controle está na mesma linha, ou está na próxima linha?

control_space_after: Há um espaço após uma palavra chave de estrutura de controle?

always_use_control_braces: As estruturas de controle sempre utilizam chaves?

else_elseif_line: Quando utilizado *else* ou *elseif*, eles foram utilizados na mesma linha, ou foram utilizados na próxima linha?

case_break_indent_from_switch: Quantas vezes o *case* e o *break* foram indentandos para abrir um bloco de *switch*?

function_space_after: As chamadas de funções possuem espaço após um nome de função e antes de abre parênteses (()?

closing_php_tag_required: Nos arquivos que somente contém PHP, o fechamento da tag `?>` é requerido?

line_endings: Que tipo de fim de linha é utilizada?

static_or_visibility_first: Qunado é feita a declaração de método, o *static* (estático) vem primeiro, ou a *visibility* vem primeiro?

control_space_parens: Na expressão da estrutura de controle, há um espaço após abre parênteses (() e um espaço antes de fecha parênteses ())? Sim = if ($expr), Não = if ($expr).

blank_line_after_php: Há uma linha em branca após abrir uma tag PHP?

class_method_control_brace: Sumário de como abre chave ({) deve ir em classes, métodos e estruturas de controles.

8.3. Resultados da Pesquisa

indent_type:
tab: 7
2: 1
4: 14

line_length_limit_soft:
?: 2
no: 3
75: 4
80: 6
85: 1

100: 1

120: 4

150: 1

line_length_limit_hard:

?: 2

no: 11

85: 4

100: 3

120: 2

class_names:

?: 1

lower: 1

lower_under: 1

studly: 19

class_brace_line:

next: 16

same: 6

constant_names:

upper: 22

true_false_null:

lower: 19

upper: 3

method_names:

camel: 21

lower_under: 1

method_brace_line:

next: 15

same: 7

control_brace_line:

next: 4

same: 18

control_space_after:

no: 2

yes: 20

always_use_control_braces:

no: 3

yes: 19

else_elseif_line:

next: 6

same: 16

case_break_indent_from_switch:

0/1: 4

1/1: 4

1/2: 14

function_space_after:

no: 22

closing_php_tag_required:

no: 19

yes: 3

line_endings:

?: 5

LF: 17

static_or_visibility_first:

?: 5

either: 7

static: 4

visibility: 6

control_space_parens:

?: 1

no: 19

yes: 2

blank_line_after_php:

?: 1

no: 13

yes: 8

class_method_control_brace:

next/next/next: 4

next/next/same: 11

next/same/same: 1

same/same/same: 6obi