PSR-1: Padrão Básico de Codificação

Essa sessão aborda sobre o que deve ser considerado como elementos padrões de codificação que assegure um alto nível de interoperabilidade entre códigos PHP compartilhados.

As palavras chaves “É OBRIGATÓRIO” (MUST) , “NÃO É OBRIGATÓRIO” (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

- Arquivos “É OBRIGATÓRIO” utilizam somente as tags `<?php` e `<?=`.
- Arquivos “É OBRIGATÓRIO” utilizem somente UTF-8 sem BOM do código PHP.
- Arquivos “DEVERIA” declara símbolos (classes, funções, constantes, etc.) ou causa efeitos colaterias (por exemplo gerar sáida (output), mudar configurações de *.ini*, etc.). Porém, “NÃO DEVERIA” realizar as duas funções (declarar símbolos ou causar efeitos colaterais) ao mesmo tempo.
- Namespaces e classes “É OBRIGATÓRIO” serem seguidos de

carregamento automático (autoloading) PSR: [PSR-0, PSR-4].

- Nomes de classes “É OBRIGATÓRIO” ser declaradas em *StudlyCaps*.
- Classes constantes “É OBRIGATÓRIO” ser declaradas em caixa alta com subtraços (_) como separadores.
- Nomes de métodos “É OBRIGATÓRIO” ser declarados em *camelCase*.

2.Arquivos

2.1. PHP Tags

Código PHP “É OBRIGATÓRIO” usar a tag longa `<?php ?>` ou a tag curta para echo `<?= ?>`; não é permitido utilizar outras variações dessas tags.

2.2. Códificação de Caracteres

Código PHP “É OBRIGATÓRIO” utilizar somente UTF-8 sem BOM.

2.3. Efeitos Colaterais

Um arquivo “DEVERIA” declara novos símbolos (classes, funções, constantes, etc.) e não causam efeitos colaterias, ou “DEVERIA” executar lógica de codificação com efeitos colaterais, mas “NÃO DEVERIA” fazer ambos.

A frase “efeitos colaterais” está relacionada com a execução de lógica (de programação) e não com declarações de classes, funções,

costantes, etc.

“Efeitos Colaterais” implementa as seguintes funções: gerar saída (output), explicita o uso de *require* ou *include*, conecta com serviços externos, modifica configurações *ini* , “ecoam” erros ou exceções, modificam variáveis globais e/ou estáticas, lêem/escrevem de/em um arquivo, etc.

Segue um exemplo de arquivo com declaração global/estática e com efeitos colaterais; esse exemplo deve ser evitado:

```
<?php
// efeito colateral: alterando configurações ini
ini_set('error_reporting', E_ALL);

// efeito colateral: carregando um arquivo
include "file.php";

// efeito colateral: gerando sáida (output)
echo "<html>\n";

// declaração
function foo()
{
// função body
}
```

Segue um exemplo de arquivo que contém declarações sem efeitos

colaterias; esse exemplo deve ser seguido:

```
<?php
// declaração
function foo()
{
// função body
}

// declaração condicional não causa um efeito colateral
if (! function_exists('bar')) {
function bar()
{
// função body
}
}
```

3.Namespace e Nomes de Classes

Namespaces e classes “É OBRIGATÓRIO” seguir o carregamento automático (autoloading) PSR: [PSR-0, PSR-4].

Isso significa que cada classe é um arquivo, e é um namespace de pelo menos do 1º level: um nome de vendedor (vendor name) de level alto.

Nomes de classes “É OBRIGATÓRIO” ser declaradas em *StudlyCaps*.

Codígo escrito em PHP 5.3 (e posterior) “É OBRIGATÓRIO” utilizar a

forma formal de namespaces.

Por exemplo:

```
<?php
// PHP 5.3 (e posterior):
namespace Vendor\Model;

class Foo
{
}
```

Código escrito em 5.2.x (e antecessor) “DEVERIA” utilizar o pseudo-namespacing, convenção de nomes de classes de “prefixos de vendedor” (Vendor_ prefixes).

```
<?php
// PHP 5.2.x (e antecessor):
class Vendor_Model_Foo
{
}
```

4.Classes Construtoras (Constants), Propriedades, e Métodos

O termo “classe” refere-se a todas as classes, interfaces e traços e/ou marcas (traits).

4.1.Construtoras (Constants)

Classes construtoras (constants) “É OBRIGATÓRIO” ser declaradas em caixa alta com subtraços (_) como separadores. Por exemplo:

```
<?php
namespace Vendor\Model;

class Foo
{
const VERSION = '1.0';
const DATE_APPROVED = '2012-06-01';
}
```

4.2. Propriedades

Este guia itencionalmente não cita nenhuma recomendação do uso de *$StudlyCaps*, *$camelCase*, ou propriedades de nomes (“property names”) de *$under_score* ($subtraço).

Qualquer que seja a convensão de nomes utilizada, “DEVERIA” ser consistentemente aplicada em um escopo (scope). Este escopo pode ser de level de vendedor (vendor-level), de level de pacote (package-level), de level de classe (class-level), ou de level de método (method-level).

4.3. Métodos

Nomes de métodos “É OBRIGATÓRIO” serem declarados em

camelCase().