Faculdades Integradas Camões

Aluno: Wendel Gonçalves de L.P

Aluno: Nikolas Eckert Nunes

Curitiba-PR-Brasil

1) *Boas Práticas para desenvolvimento PHP*

Quando estamos desenvolvendo nosso código PHP, ninguém nos define regras de como desenvolver. Podemos fazer como quisermos e é bom que a gente tenha essa liberdade.

Mas, na medida que nosso sistema cresce e começamos a implementá-lo em vários lugares, surge, inevitavelmente, a necessidade de seguir algum padrão para que quem vá implementar ou dar manutenção consiga entender o que está acontecendo.

2) PSRs ou Guia de Estilo de Codificação

PSR é o acrônimo para PHP Standards Recommendations (que em português pode ser traduzido para: Recomendações de padrões para PHP).

Pelo próprio nome já dá para ter uma ideia do isso significa, nada mais é que um manual de recomendações para o desenvolvimento com PHP. Ou seja, as PSRs tem o objetivo de definir padrões de códigos para que a comunidade siga e aplique as boas práticas da linguagem.

3) Práticas recomendadas de codificação

Uma das PSRs é a PSR 2, ou Guia de estilo de codificação (Coding Style Guide), que aborda como deve ser feita a formatação do nosso código para facilitar a leitura por outros desenvolvedores. Algumas das indicações desse guia são: Devemos usar 4 espaços para identação, não tabs.

- *Não devemos fixar um número de caracteres por linha, mas é bom que uma linha tenha menos de 80 caracteres.*

- *A abertura de colchetes de classes e métodos devem vir na próxima linha.*

- *A abertura de colchetes de estruturas de controle deve vir na mesma linha com um espaço em branco.*

Se seguirmos a PSR 2 com nosso método, teríamos:

```
class Validator
{
    public function validaUsuario($usuario)
    {
        if (self::validaNome($usuario->getNome())
            && self::validaEmail($usuario->getEmail())
            && self::validaSenha($usuario->getSenha())) {
                return true;
        } else {
            return false;
        }
    }
}
```

4) Visão Geral

PSRs não servem apenas para identação de código. Quando trabalhamos com frameworks MVC é muito comum precisarmos importar arquivos. Algo como:

```
include_once "arquivo.php";
```

Porém conforme o sistema cresce, precisamos importar mais arquivos:

```
include_once "Namespace/SubNamespace/arquivo.php";
include_once "Namespace/SubNamespace/arquivo2.php";
include_once "Namespace/SubNamespace/arquivo3.php";
include_once "Namepsace/SubNamespace/arquivo4.php";
```

E a quantidade de includes tende a crescer. Esse é um dos motivos pelo qual utilizamos a funcionalidade de `autoload`! Existem diversas formas de se realizar `autoload` em php, uma delas é a ***PSR-4 que define uma forma padronizada de escrever nossos*** *`namespace` para que possamos importar nossos arquivos da mesma forma em nossos sistemas, aumentando, também, a interoperabilidade entre projetos! Para isso, declaramos nossos namepsace com a estrutura:*

```
namespace Namespace\SubNamespace;
```

Agora para importar nossos arquivos, a partir da versão 7 do PHP, basta utilizarmos o agrupamento do use:

```
use Namespace\SubNamespace\{arquivo1,arquivo2,arquivo3,arquivo4};
```

Alguns frameworks MVC estão implementando este padrão para realizar o ***autoLoad****. Um exemplo é o Laravel, a partir da versão 5.* Atualmente existem 14 PSRs criadas pela PHP-FIG *(Framework Interoperability Group)sendo que estão em desenvolvimento e cinco já foram aceitas.*

5) Referências

https://blog.especializati.com.br/php-psrs/

http://blog.alura.com.br/boas-praticas-de-desenvolvimento-php/

https://www.turbosite.com.br/blog/30-melhores-praticas-em-php-para-iniciantes

https://forum.imasters.com.br/topic/214963-boas-pr%C3%A1ticas-ao-programar/