

Faça o que eu fiz na aula

Nas últimas aulas, para chamarmos as outras classes e podermos utilizá-las no arquivo `index.php`, nós utilizamos a palavra-chave `require`, e como carregamos dois arquivos, precisamos chamá-lo duas vezes:

```
require 'ArrayUtils.php';  
require 'Calculadora.php';
```

E se tivermos que carregar mais arquivos e classes, teríamos que adicionar uma linha de `require` para cada um. Se quisermos adicionar automaticamente todas as classes que criarmos, como podemos fazer?

O PHP possui uma funcionalidade chamada Autoload, que permite o carregamento automático de classes. Desse jeito, só chamamos o autoloader uma vez, e depois disso, todas as classes serão carregadas automaticamente.

Antes de construirmos o autoload, é importante mencionar que seguiremos o padrão PSR-4, criado pelo grupo PHP-FIG (Framework Interop Group) que é um grupo de pessoas e projetos da comunidade PHP que definem o que chamamos de PSR (PHP Standards Recommendations), que são convenções bem definidas para aplicarmos em nosso código. E para isso, vamos primeiramente adicionar, nas classes que temos criadas, um namespace.

O namespace serve para separar as classes do nosso projeto das demais. Imagine que temos no nosso projeto uma classe chamada Mail, e importarmos uma biblioteca que possui uma classe com este mesmo nome, como o PHP sabe que classe ele deve usar? O namespace nos ajuda a diferenciar as classes quando algo deste tipo acontece.

Vamos criar uma pasta chamada `src`, e dentro dela uma pasta chamada `Alura`, e vamos mover os arquivos de classes que temos criados (`ArrayUtils.php` e `Calculadora.php`) para dentro dela.

Nós vamos utilizar a função `spl_autoload_register`, essa função nos permite registrar uma função de autoload, e ela aceita uma como argumento.

```
spl_autoload_register(  
    function (string $namespaceClasse): void {  
  
    }  
);
```

Essa função que utilizaremos como argumento aceitará um namespace da classe como argumento, então colocamos ele.

Agora dentro desta função, vamos definir o diretório desta classe dentro do nosso projeto, vamos criar uma variável para armazenar o diretório raiz:

```
spl_autoload_register(  
    function (string $namespaceClasse): void {  
        $caminho = "/src";  
    }  
);
```

Agora vamos encontrar o diretório da classe, como ele segue o mesmo padrão de nomes do namespace, vamos utilizar a seguinte função:

```
spl_autoload_register(  
    function (string $namespaceClasse): void {  
        $caminho = "/src";  
        $diretorioClasse = str_replace("\\", DIRECTORY_SEPARATOR, $namespaceClasse);  
    }  
);
```

Essa função vai pegar a barra invertida do namespace que temos que escapar com mais uma barra \ e trocar ela pela constante `DIRECTORY_SEPARATOR` que possui o separador de diretório padrão do sistema operacional que estamos utilizando.

Por fim, chamamos uma linha de include juntando tudo:

```
spl_autoload_register(  
    function (string $namespaceClasse): void {  
        $caminho = "/src";  
        $diretorio_classe = str_replace("\\", DIRECTORY_SEPARATOR, $namespaceClasse);  
        @include_once getcwd() . $caminho . DIRECTORY_SEPARATOR . "{$diretorio_classe}.php";  
    }  
);
```

Para separarmos o código do autoloader do nosso arquivo `index`, vamos copiar este código e colar em um arquivo novo, chamado `autoload.php`

```
spl_autoload_register(  
    function (string $namespaceClasse): void {  
        $caminho = "/src";  
        $diretorio_classe = str_replace("\\", DIRECTORY_SEPARATOR, $namespaceClasse);  
        @include_once getcwd() . $caminho . DIRECTORY_SEPARATOR . "{$diretorio_classe}.php";  
    }  
);
```

E agora no arquivo `index.php`, usamos o `require` para chamar o autoloader.

```
require 'autoload.php';
```

E só precisaremos desse `require` na aplicação toda. Agora podemos chamar as classes que criamos usando a seguinte sintaxe:

```
$calculadora = new Calculadora();  
  
var_dump($calculadora);
```

Toda vez que utilizarmos a palavra-chave `new` no PHP, o PHP vai chamar a função `spl_autoload_register` e vai carregar para a classe que queremos, sem precisar do `require`.

