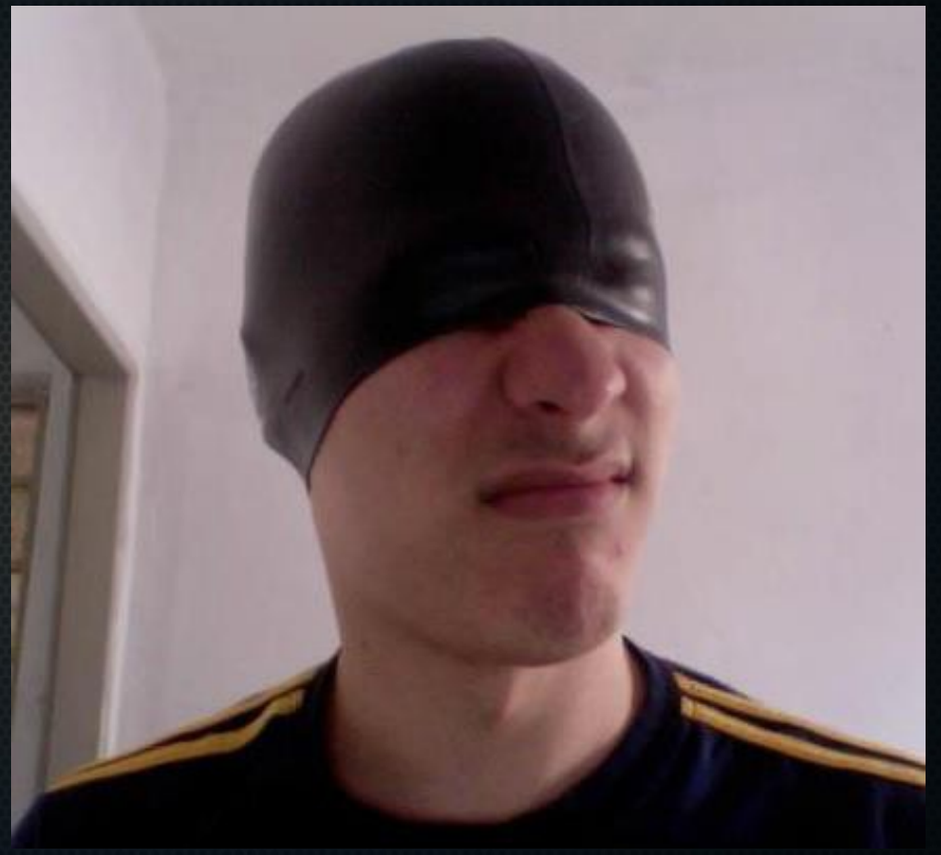
PHP-CLI em 7 passos

Henrique Moody

- Desenvolvedor web com foco em PHP desde 2007, usuário assíduo de Linux desde 2008 e Zend Certified Engineer desde 2011.
- Atua com Desenvolvedor e Líder Técnico na Dafiti - empresa de comércio eletrônico brasileira.
- Forte entusiasta da comunidade PHP no Brasil e contribuidor de projetos Open Source como Respect (https://github.com/Respect) e Composer (https://github.com/composer) dentre vários outros.



http://about.me/henriquemoody

PHP nasceu para web

- A maior parte das aplicações PHP é escrita para rodar em ambiente web;
- Nasceu sob o nome the *Personal Home Page/Forms Interpreter*;
- Após o lançamento da versão 4 houve a necessidade de rodar PHP fora da web.

Bilú quer a concha!

CLI - Command-Line Interface

- PHP suporta CLI desde o PHP 4.3.0.
- O foco principal deste SAPI é para o desenvolvimento de aplicações Shell (concha) com PHP;
- Funciona em qualquer OS.

Sete passos!

1) Habilitar PHP-CLI

Um passo

Habilitando PHP-CLI

- Vem habilitado por padrão na maioria das distrubições Linux;
- Para habilitar ao compilar utilize `--enable-cli` (habilitado por padrão);
- Windows: verifique a variável de ambiente PATH;
- Debian-based: `apt-get install php-cli`;
- RPM-based: `yum install php-cli`.

1) Habilitar PHP-CLI;
2) Shebang

Dois passos

Shebang

- Um shebang refere-se aos dois caracteres "#!", quando os mesmos são os primeiros caracteres de um arquivo de texto, especificamente em um código fonte escrito em uma linguagem interpretada.
- O sistema tenta executar o arquivo usando um interpretador especificado pelo shebang.

Exemplo

- Scripts em bash iniciam-se com o shebang

```
#!/bin/bash
echo Hello
```

Exemplo

- Scripts em PHP iniciam-se com o shebang

```
#!/bin/php
<?php
echo "Hello";
```

Path do interpretador

- Algumas vezes o interpretador pode estar em um path diferente, por exemplo, o PHP pode estar em `/usr/bin/php` e não em `/usr/bin/php`.
- Para resolver este problema podemos usar o comando `/usr/bin/env` que define as variáveis de ambiente.

script.php

```
#!/usr/bin/env php
<?php
echo 'Hello, ' . $_SERVER['USER'] . '!' . PHP_EOL;
```

Terminal

```
$ chmod +x script.php
$ ./script.php
Hello, henriquemoody!
```

Três passos

1) Habilitar PHP-CLI;
2) Shebang;
3) Passagem de argumentos

$argv

- Array com os valores dos argumentos passados.

script.php

```
#!/usr/bin/env php
<?php
print_r($argv);
```

Terminal

```
$ ./script.php Henrique Moody
Array
(
    [0] => ./script.php
    [1] => Henrique
    [2] => Moody
)
```

$argc

- Número de parâmetros passados

script.php

```
#!/usr/bin/env php
<?php
print_r($argc);
```

Terminal

```
$ ./script.php
3
```

$argc + $argv

- Combinação poderoza que permite parsear os argumentos passados

script.php

```
#!/usr/bin/env php
<?php
$options = array();
for ($i=1; $i < $argc; $i=$i+2) {
    $key = $argv[$i];
    $value = $argv[$i+1];
    $options[$key] = $value;
}
print_r($options);
```

Terminal

```
$ ./script.php --first Henrique --last Moody
Array
(
    [--first] => Henrique
    [--last] => Moody
)
```

`getopt()`

- Facilita o parser de argumentos

script.php

```php
#!/usr/bin/env php
<?php
$shortopts  = "";
$shortopts .= "f:";  // Valor obrigatório
$shortopts .= "v::"; // Valor opcional
$shortopts .= "abc"; // Opções que não aceitam valores

$longopts  = array(
    "required:",     // Valor obrigatório
    "optional::",    // Valor opcional
    "option",        // Sem valor
    "opt",           // Sem valor
);
$options = getopt($shortopts, $longopts);
print_r($options);
```

Terminal

```
$ ./script.php -f "value for f" -v -a --required value --optional="optional value" –opt
Array
(
    [f] => value for f
    [v] =>
    [a] =>
    [required] => value
    [optional] => optional value
    [option] =>
)
```

Bilú que objeto!

Pacotes PHP para CLI scripts

- Console_Getopt (PEAR);
- Zend\Console;
- Symfony\Component\Console.

Quatro passos

1) Habilitar PHP-CLI;
2) Shebang;
3) Passagem de argumentos;
4) STDIN

STDIN

- Entrada padrão de dados.

script.php

```
#!/usr/bin/env php
<?php
echo "Digite o seu nome: ";
$name = fgets(STDIN);
print_r($name);
```

Terminal

```
$ ./script.php
Digite o seu nome: Henrique Moody
Henrique Moody
```

Cinco passos

1) Habilitar PHP-CLI;
2) Shebang;
3) Passagem de argumentos;
4) STDIN;
5) STDOUT

STDOUT

- Saída padrão de dados.

script.php

```
#!/usr/bin/env php
<?php
fwrite(STDOUT, 'Hello, ' . $_SERVER['USER'] . '!' . PHP_EOL);
```

Terminal

```
$ ./script.php
Hello, henriquemoody!
```

Seis passos

1) Habilitar PHP-CLI;
2) Shebang;
3) Passagem de argumentos;
4) STDIN;
5) STDOUT;
6) STDERR

STDERR

- Saída de erro padrão.

script.php

```
#!/usr/bin/env php
<?php
fwrite(STDERR, 'Hello, ' . $_SERVER['USER'] . '!' . PHP_EOL);
```

Terminal

```
$ ./script.php
Hello, henriquemoody!
```

What dá Fuck?

Relaxe os tendões!

STDOUT e STDERR

- Para um usuário aparentemente não há diferença, mas para um programa há.

Terminal

```
$ ./stdout.php
Hello, henriquemoody!
$ ./stderr.php
Hello, henriquemoody!
```

Terminal

```
$ ./stderr.php > /dev/null
Hello, henriquemoody!
$ ./stdout.php > /dev/null
```

Terminal

```
$ ./stdout.php 2> /dev/null
Hello, henriquemoody!
$ ./stderr.php 2> /dev/null
```

Sete passos

1) Habilitar PHP-CLI;
2) Shebang;
3) Passagem de argumentos;
4) STDIN;
5) STDOUT;
6) STDERR
7) Exit code

Exit code

- Exit code, ou return code de um processo é um número inteiro que é passado do processo filho para o processo pai quando o processo filho termina sua execução.
- O número deve ser entre 0-255, ao forçar um número fora desse intervalo o comportamento é intermitente;
- 0 é considerado sucesso;
- Qualquer número maior que 0 é considerado erro;
- O programador pode/deve documentar os tipos de erro por número.

script.php

```
#!/usr/bin/env php
<?php
fwrite(STDOUT, 'Quanto Ãé 1 + 1? ');
$value = (int) fgets(STDIN);
if ($value === 2) {
    fwrite(STDERR, 'Certa resposta!' . PHP_EOL);
    exit(0);
} else {
    fwrite(STDOUT, 'Resposta errada, manolo!' . PHP_EOL);
    exit(1);
}
```

Terminal

```
$ ./script.php
Quanto é 1 + 1? 2
Certa resposta!
$ echo $?
0
```

Terminal

```
$ ./script.php
Quanto é 1 + 1? 42
Resposta errada, manolo!
$ echo $?
1
```

Perguntas?

Recursos

- Exemplos da palestra: https://gist.github.com/fd9a8187f410c03bebb1;
- PHP-CLI: http://php.net/cli;
- PHP `getopt()`: http://php.net/getopt;
- Console_Getopt: http://pear.php.net/Console_Getopt;
- Zend\Console: http://goo.gl/SchES;
- Symfony\Console: https://github.com/symfony/Console.

Busquem conhecimento