$ building for the command line inter

Steve Grunwell
Staff Software Engineer, Mailchim

@stevegrunwell@phpc.social
stevegrunwell.com/slides/php-c

WHY THE CLI?

PHP EVERYWHERE!

- Re-use application cod
- Reduce language sprav
- PHP ❤️Scripting

INVOKING PHP ON THE

Via the PHP binary:

```
$ php my-command.php
```

With the PHP shebang:

```
#!/usr/bin/env php
```

```
$ chmod +x my-command.php
$ ./my-command.php
```

WHEN MIGHT I USE THE

- Data migrations & transform
- Maintenance scripts
- Dev-only actions
  - Scaffolding
  - Other code changes
- "#YOLO scripts"

CLIs FOR YOUR FAVO
FRAMEWORKS

DRUSH

- "Drupal Shell"
  - One of the OG CLI tools for PHP
- Manage themes, modules, system

WP-CLI

- Install core, themes, plugins, etc.
- Manage posts, terms, users, and m
- Inspect and maintain cron, caches transients
- Extensible for themes + plugins

LARAVEL ARTISAN

- The underlying CLI for Laravel
  - Built atop the Symfony Console
- Scaffold 
- Allows packages to register new

JOOMLATOOLS CONSO

- CLI framework for Joomla
- Manage sites, extensions, datab
  - Includes virtual host manage

CLI CONCEPTS

COMPOSABILITY

Good CLI commands should be **co**

RULE OF COMPOSABIL

> *Developers should write program*
> *communicate easily with other p*
> *This rule aims to allow developer*
> *down projects into small, simple*
> *rather than overly complex mo*
> *programs.*

—Eric S. Raymond, *The Art of Unix Prog*

DATA STREAMS

Three default data stream

0. STDIN - input
1. STDOUT - output
2. STDERR - errors

DATA STREAMS IN PRAC

```
# Get the number of unique IP addresses in acc
$ grep -Eo "([0-9]{1,3}[\.]){3}[0-9]{1,3}" \
    /var/log/nginx/access.log \
    | uniq \
    | wc -l \
    | xargs printf "%d unique IP addresses det
43282 unique IP addresses detected
```

EXIT CODES

Exit codes tell us how everythin

Code	Meaning
0	All good!
1	Generic error
2	Incorrect command/ar
3–255	Specific errors

EXIT CODES & BOOLEAN OPE

```
# Celebrate a non-zero exit code!
$ do-something && celebrate

# Hang your head in shame if something fails
$ do-something || hang-head-in-shame

# Put the operators together
$ (do-something && celebrate) || hang-head-in-

# Semi-colons don't care, they just separate c
$ do-something; celebrate; hang-head-in-shame
```

ARGUMENTS + OPTION

```
# Arguments
$ cd /var/www
$ grep "Some text" file.txt

# Options
$ git commit -m "This is my commit message"
$ ls -a -l
$ ls -al

# Long options
$ composer outdated --format=json
$ git push --force-with-lease
```

CONVENTIONS FOR OPTI

```
OPTIONS:

  -h|--help     Print usage instructions
  -q|--quiet    Silence all output
  -v|--version  Print version information
  --verbose     Print additional output
```

ENVIRONMENT VARIABL

Set and read variables in the c
environment

```
# Export from shell files
export CURRENT_CITY="Bowling Green"

# Set directly in shell
$ CURRENT_CITY="Chicago"

# Set as you call a command
$ CURRENT_CITY="Rosemont" some-script
```

ENVIRONMENT VARIABLES

```
# Get array of all environment variables
getenv();

# Retrieve a specific variable (false if unset
getenv('SOMEVAR');

# Set an environment variable
putenv('SOMEVAR=some_value');

# Delete an environment variable
putenv('SOMEVAR=');
```

THE CLI SAPI

Additional **S**erver **API** for P

```
// Check the current SAPI. We can also use PHP
if (php_sapi_name() === 'cli') {
    // We're on the command line!!
}
```

SPECIAL CLI GLOBALS

int **$argc**

Argument count

array **$argv**

Argument values

Both will always have at least on

WHAT WILL WE SEE?

```
$ php -r 'echo "{$argc} arg(s):\n"; var_export
    PHP "is great"
```

```
3 arg(s):
array (
  0 => 'Standard input code',
  1 => 'PHP',
  2 => 'is great',
)
```

DAEMONS

A process that continually runs
background

```
while (true) {
    // do something!
}
```

Building PHP Daemons and
Running Processes

WRITING CLI COMMA

github.com/stevegrunwell/php-cli

A SIMPLE GREETER

```
#!/usr/bin/env php
<?php

$name = $argv[1] ?? 'there';

printf("Hello, %s!\n", $name);
```

```
$ php hello.php Ben
Hello, Ben!
```

```
$ php hello.php
Hello, there!
```

ACCEPTING OPTIONS

```
#!/usr/bin/env php
#
# USAGE:
#
#     hello.php [-g|--greeting=<greeting>] <na
<?php

$opts = getopt('g:', [
    'greeting:',
], $index);
$greeting = $opts['greeting'] ?? $opts['g'] ??
$name = $argv[$index] ?? 'there';

printf("%s, %s!\n", $greeting, $name);
```

```
$ php hello.php --greeting="Salutations" Dylan
Salutations, Dylan!
```

```
$ php hello.php -g="Salutations" Dylan
Salutations, Dylan!
```

WE CAN DO BETTER THAN ge

PERFORMING SYSTEM OPER

- PHP has built-in functions for thin
  `chmod()`, `mkdir()`, etc.
  - Even more with Flysystem
- Can also execute arbitrary system

CALLING OTHER SCRIP

`exec()`

Execute, return the last line of
Can capture full output as array,

`shell_exec()`

Execute, return the full output a

CALLING OTHER SCRIP

`system()`

Returns last line of outpu

Flushes buffer as it goes

`passthru()`

Best choice for binary file

ESCAPING COMMANDS & ARG

`escapeshellcmd()`

Escape an entire comman

`escapeshellarg()`

Escape an individual argum

WITHOUT ESCAPING

```
$name = 'Larry && rm -rf /';

# Uh oh, $name isn't being escaped!
exec('greet-user ' . $name);
```

```
# You're about to have a very bad day...
Hello, Larry!
```

WITH PROPER ESCAPI

```php
$name = 'Larry && rm -rf /';

# Escape the argument with escapeshellarg()
exec('greet-user ' . escapeshellarg($name));
```

```
# Weird name, but no harm done
Hello, Larry && rm -rf /!
```

LIBRARIES & FRAMEW

SYMFONY CONSOLE

- CLI framework of choice
- Handlers for input & outpu
- Built-in help screen, valida
- Born to be tested

BUILDING A SYMFONY CONSOLE CO

```php
namespace App\Command;

use Symfony\Component\Console\Attribute\AsComm
use Symfony\Component\Console\Command\Command;

#[AsCommand(name: 'app:create-user')]
class CreateUserCommand extends Command
{
    // ...
```

CONFIGURING THE COMMAN

```
protected function configure(): void
{
    $this->setDescription('Creates a new user.
        ->setHelp(/* Full help text goes here.
        ->addArgument(/* ... */)
        ->addOption(/* ... */);
}
```

THE EXECUTE() METHOD

```
use Symfony\Component\Console\Input\InputInter
use Symfony\Component\Console\Output\OutputInt

protected function execute(
    InputInterface $input,
    OutputInterface $output
 ): int {
    // Do something in here!

    return Command::SUCCESS;
}
```

ARGUMENTS + OPTIONS

```
$user = new User($input->getArgument('email'));

if ($input->getOption('admin')) {
    $user->makeAdmin();
}

$user->save();
```

BOOTSTRAP OUR COMMAND

```
#!/usr/bin/env php
<?php

require __DIR__ . '/vendor/autoload.php';

use App\Command\CreateUserCommand;
use Symfony\Component\Console\Application;

$app = new Application();
$app->add(CreateUserCommand());
$app->run();
```

CALLING OUR COMMAND

```
$ php console.php app:create-user beth@example

# If we've made console.php executable
$ console.php app:create-user andy@example.com

# Produce the help documentation
$ php console.php app:create-user --help
```

PHP-CLI TOOLS

- Maintained by the WP-CLI t
- Simplify input + output
  - Prompts, menus, and mor
  - Output formatting: tables, progress bars, and more!

PHP-CLI TOOLS

```
#!/usr/bin/env php
<?php

require_once __DIR__ . '/vendor/autoload.php';

$limit  = cli\prompt('How high should I count?
$loud   = cli\choose('Shall I shout it');
$suffix = $loud === 'y' ? '!' : '.';

for ($i = 1; $i <= $limit; $i++) {
    cli\line($i . $suffix);
}
```

PHP-CLI TOOLS

```
$ php Counter.php
How high should I count? [10]: 5
Shall I shout it? [y/N]y
1!
2!
3!
4!
5!
```

CLImate

- The League of Extraordinary Pack
- More focused on output
  - Progress bars, borders, JSON, an
- Includes helpers for ASCII art and

CLI BEST PRACTIC

CHECK YOUR ASSUMPTI

- Check that commands exist before
- Don't hard-code system paths

RULE OF SILENCE

> *Developers should design progra*
> *they do not print unnecessary ou*
> *rule aims to allow other progra*
> *developers to pick out the inform*
> *need from a program's output*
> *having to parse verbosity*

—Eric S. Raymond, *The Art of Unix Prog*

```
# Default behavior
$ some-command
Command completed successfully!

# Only produce output if something went wrong
$ some-command --quiet

# Be more verbose
$ some-command --verbose
Reindexing database...OK
Reticulating splines...OK
Command completed successfully!
```

GARBAGE COLLECTIO

- Clean up objects when you're
- Be judicious with caching
- Watch for ballooning objects &

IGNORE WEB REQUEST

If your commands live within the
prevent them from being run outsi

```
// Only allow this script to run on the CLI!
if (PHP_SAPI !== 'cli') {
    exit;
}
```

SWANSON ON COMMAN

THANK YOU!

Steve Grunwell
Staff Software Engineer, Mailchim

@stevegrunwell@phpc.social
stevegrunwell.com/slides/php-c
github.com/stevegrunwell/php-cli-exa