BUILDING FOR THE PHP COMMAND LINE INTERFACE

Steve Grunwell
@stevegrunwell

stevegrunwell.com/slides/php-cli

WHO AM I?

- Senior Software Engineer @ Liquid Web
- Open-source contributor
- Husband + (new) father
- Coffee roaster + homebrewer

@stevegrunwell - stevegrunwell.com/slides/php-cli

WHY THE CLI?

BENEFITS

- Cut the browser cord
- Speak your app's language
- No need to learn Bash!

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

HOW DOES IT RUN?

Invoking PHP on the CLI:

```
# 1. Pass the file to the php binary
$ php my-command.php
```

```
# 2. Execute the file directly if using the php shebang.
$ chmod +x my-command.php
$ ./my-command.php
```

WHEN MIGHT I USE THEM?

- Data migrations & transformations
- Maintenance scripts
- Actions that should only be available via a console
  - Scaffolding
  - Other code changes
- "#YOLO scripts"

BEST PRACTICES

COMPOSABILITY

Good CLI commands should be **composable!**

RULE OF COMPOSITION

Developers should write programs that can communicate easily with other programs. This rule aims to allow developers to break down projects into small, simple programs rather than overly complex monolithic programs.

DON'T ASSUME ANYTHING!

- Your script could be run on a wide variety of systems
- Check that system commands work before calling them

KNOW THE CURRENT SERVER API

```php
// Make sure this script is being run over the PHP CLI!
if ('cli' !== php_sapi_name()) {
    return;
}
```

RULE OF SILENCE

Developers should design programs so that they do not print unnecessary output. This rule aims to allow other programs and developers to pick out the information they need from a program's output without having to parse verbosity.

GARBAGE COLLECTION

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

IN THE WILD

DRUSH

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

WP-CLI

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

ARTISAN

- The underlying CLI for Laravel
- Scaffold 
- Built atop the Symfony Console
- Third-party access

JOOMLA CONSOLE

- CLI framework for Joomla!
- Manage sites, extensions, databases, & virtual hosts
- List, purge, and clear cache contents

WRITING YOUR OWN SCRIPTS

SYMFONY CONSOLE COMPONENT

- CLI framework of choice
- Handlers for input & output
- Born to be tested

PHP-CLI TOOLS

- Simplify input + output
  - Tabular + tree displays
  - Progress indicators
- Maintained by the WP-CLI team

CLImate

- More focused on output
  - Progress bars, borders, JSON, and more
- Includes helpers for ASCII art and animations!

TYPES OF INPUT

[Positional] Arguments

```
$ my-command foo bar
```

Options ("associative args")

```
$ my-command -n=100 --type foo --verbose
```

$argv

- Arguments passed to script as array
- Includes script name!

```
$ php -r "var_export(\$argv);" foo bar

array (
    0 => 'Standard input code',
    1 => 'foo',
    2 => 'bar',
)
```

$argc

- # of arguments
- Also includes script name!

```
$ php -r "var_export(\$argc);" foo bar

3
# Script name, "foo", and "bar"
```

getopt()

```php
# myscript.php
var_export(getopt(
    'a:bc',
    ['foo:', 'verbose']
));
```

```
$ myscript -a=hello -b -d --foo=bar --verbose

array(
  'a' => 'hello'
  'b' => false,
  'foo' => 'bar',
  'verbose' => false,
)
```

SYSTEM COMMANDS

- PHP can natively do a lot of common Unix operations: `copy`, `chmod`, `chown`, etc.
- Can also execute arbitrary system operations!

MY SECURITY-SENSE IS TINGLING!

EXECUTING SYSTEM COMMANDS

`exec()` Returns the last line of output as a string.

`shell_exec()` Returns full output as a string.

Same as backtick operator:

```
`ls -al` === shell_exec('ls -al')
```

(EVEN MORE) SYSTEM COMMANDS

`system()`	Flush the output buffer after each line.
`passthru()`	Returns the raw output.
	Great for working with binary files or interactive output!

ESCAPING SHELL COMMANDS

`escapeshellcmd()`

Escapes any meta-characters that could be used to execute arbitrary commands

`escapeshellarg()`

Escape individual command arguments.

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

# Oh no, $name isn't being escaped!
exec('greet-user ' . $name);
```

```
> Hello, steve # proceeded by your system being destroyed
```

MUCH BETTER:

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

# Nice try, user!
exec('greet-user ' . escapeshellarg($name));
```

```
> Hello, steve && rm -rf / # What an odd name!
```

ENVIRONMENT VARIABLES

```php
$name = getenv('DEMO_NAME');

if ($name) {
    printf('Hey, I recognize you, %s!' . PHP_EOL, $name);
    $name = sprintf('My old friend, %s!', $name);


} else {
    echo "I don't know you, so I'll just call you Fred." . PHP_EOL;
    $name = 'Fred';
}

// Update DEMO_NAME and call the system's echo program.
putenv('DEMO_NAME=' . $name);
passthru('echo $DEMO_NAME');
```

ENVIRONMENT VARIABLES

```
export DEMO_NAME="Steve"
```

```
> Hey, I recognize you, Steve!
> My old friend, Steve!
```

Otherwise:

```
> I don't know you, so I'll just call you Fred.
> Fred
```

EXIT CODES

- The way we exit scripts is significant:
  - `0` = successful
  - 1 = error
  - 2-255 = special meaning
- Will always use last exit code

EXIT CODES

```php
if (! isset($argv['1'])) {
    echo "Missing required argument!";
    exit(1);
}


// Do something awesome
```

```
$ php my-script.php foo && echo "Success"
$ php my-script.php && echo "You will never see this"
```

DAEMONS

A process that continually runs in the background

```
while ( $run ) {
    // do something!
}
```

EXAMPLES

github.com/stevegrunwell/php-cli-examples

SYMFONY CONSOLE

```php
<?php
use Symfony\Component\Console\Command\Command;
use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputInterface;
use Symfony\Component\Console\Output\OutputInterface;

class SymfonyExample extends Command
{
    // ...
```

SYMFONY CONSOLE

```php
/**
 * Define what the command should be called and what arguments it
 * should accept.
 */
protected function configure()
{
    $this
        ->setName('symfony-example')
        ->setDescription('Greet a user by name.')
        ->addArgument(
            'name',
            InputArgument::REQUIRED,
            'The name of the user'
        );
}
```

SYMFONY CONSOLE

```php
/**
 * Execute the command.
 *
 * @param InputInterface  $input  The input interface.
 * @param OutputInterface $output The output interface.
 */
protected function execute($input, $output)
{
    $output->writeln(sprintf(
        '<comment>Symfony says "hello", %s!</comment>',
        $input->getArgument('name')
    ));
}
```

SYMFONY CONSOLE

```
$ php examples/SymfonyExample.php symfony-example Steve
```

```
> Symfony says "hello", Steve!
```

PHP-CLI TOOLS

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

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


$limit  = cli\prompt('How high should I count?', 10);
$loud   = cli\choose('Shall I shout it');
$suffix = 'y' === $loud ? '!' : '.';

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

PHP-CLI TOOLS

```
$ php examples/PHPCliToolsExample.php
```

```
$ php examples/PHPCliToolsExample.php
How high should I count? [10]: 5

Shall I shout it? [y/N]y
1!
2!
3!
4!
5!
```

WP-CLI: REGISTER COMMAND

```php
/**
 * Example WP-CLI script.
 */

class Example_WP_CLI_Command extends WP_CLI_Command {
    // At least one public method.
}

WP_CLI::add_command( 'example-command', 'Example_WP_CLI_Command' );
```

WP-CLI: COMMAND DEFINITION

```
/**
 * Display the latest posts from a given user.
 *
 * ## OPTIONS
 *
 * <login>
 * : The user login to collect stats for.
 *
 * @subcommand latest-posts-by-user
 */
public function latest_posts_by_user( $args, $assoc_args ) {
    // Do something!
}
```

WP-CLI: COMMAND METHOD

```php
public function latest_posts_by_user( $args, $assoc_args ) {
    $user = get_user_by( 'login', $args['0'] );
    if ( ! $user ) {
        return WP_CLI::error(
            'The specified user login does not exist!'
        );
    }

    $posts  = get_posts( array( 'author' => $user->ID ) );
    $fields = array( 'ID', 'post_title', 'post_date' );

    return WP_CLI\Utils\format_items( 'table', $posts, $fields );
}
```

WP-CLI: IN ACTION

```
$ wp example-command latest-posts-by-user admin
```

```
+-----+---------------+---------------------+
| ID  | post_title    | post_date           |
+-----+---------------+---------------------+
| 7   | A third post  | 2016-04-08 14:32:00 |
| 5   | Another post  | 2016-04-05 18:32:23 |
| 1   | Hello World!  | 2015-11-12 01:14:38 |
+-----+---------------+---------------------+
```

THANK YOU!

Steve Grunwell
stevegrunwell.com
liquidweb.com

stevegrunwell.com/slides/php-cli