Phar, The PHP .exe format

Helgi Þormar Þorbjörnsson

PHP Benelux, 27th of January 2012

Who am I?

Helgi

Co-founded Orchestra.io

Work at EngineYard

PEAR Developer

From Iceland

@h on Twitter

What is Phar?

Phar

=

PHP Archive

Similar to JAR

●●● But for PHP

Phar

PHP Extension

Built in / default from 5.3 onwards

More Powerful than PHP_Archive

PEAR: PHP_Archive

Reference implementation

User Land Code

Less Powerful

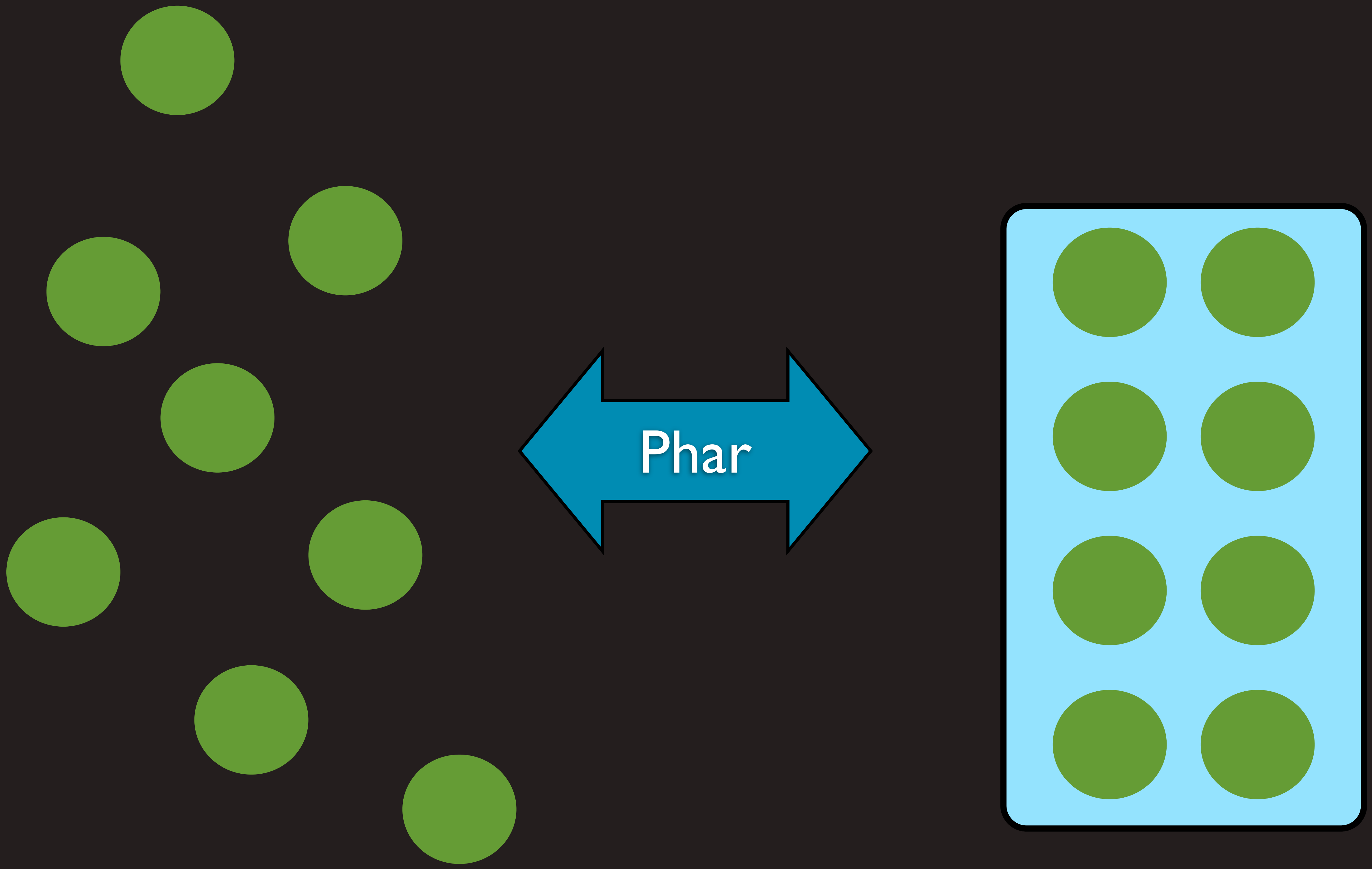
Works on older PHP

Not maintained anymore

Phar unravelled

Simple yet flexible File Format

Pack multiple files into one


Phar

Similar to Tar

The special sauce

1. Stub
2. Manifest
3. File Contents
4. Signature (optional)

Manifest

List of Files

File Permission

File Compression

Meta Data

Compression

Whole Archive

Per file

Available Compressions

gzip, bz2 and zip

Stub

Piece of PHP code

Bootstraping

Autoloader

Get all the code used in this talk and more!

https://github.com/helgi/phar-talk

Going more practical

Normal Include

```php
include 'awesome.php';
```

Phar Include

```
include 'phar://example2/project.phar/awesome.php';
```

Phar works via Streams

Streams Usage

fopen / fwrite / fclose

file_get_contents

opendir / rmdir / mkdir

anything that works with streams

```php
header('Content-type: text/javascript');
echo file_get_contents('phar://example3/project.phar/js/zepto.js');
```

Does not extract to disk

Works on web and CLI

Can be ran without the extension

Why use Phar?

Full Applications

CLI Applications

Pyrus

PHPUnit

Web Applications

Setup Tools

go-pear.phar

PEAR installation in PHP builds

Supporting Code

Plugins

Lithium

Themes

Libraries

Silex

Handy Packaging!

Executable file for applications!

Pros and Cons

Pros

Single download with all dependencies

Run multiple versions in parallel

Upgrades are easy

No unpacking

Security against modifications

Cons

Incremental updates (no deltas)

Upgrading is a manual process

Web server may need changes

Extending the application is harder

README / INSTALL become hard to reach

Stubs

Bootstrap for phar

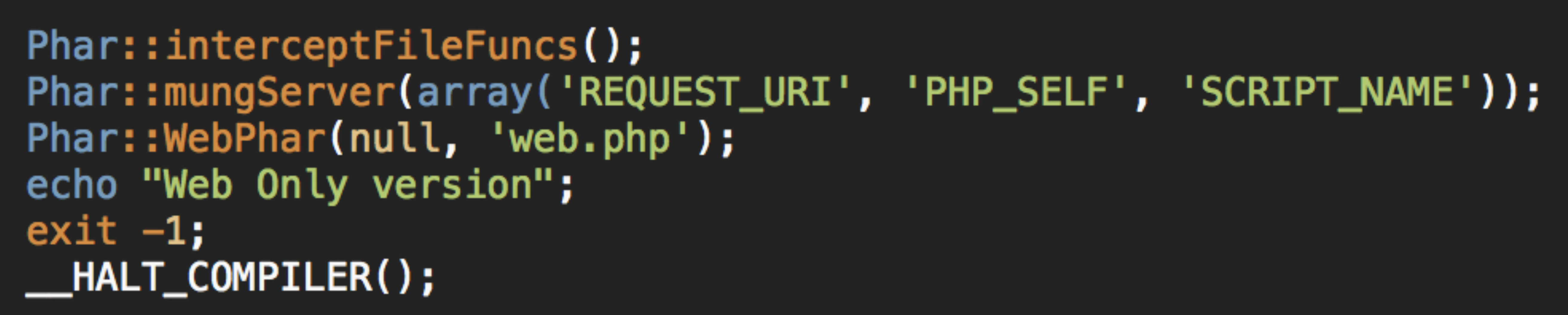
Smallest possible Stub

```
$phar = new Phar('project.phar');
$phar->setStub('<?php __HALT_COMPILER();');
```

__HALT_COMPILER() is a **minimum** requirement

Not used when phar file is used via streams

Web Phar



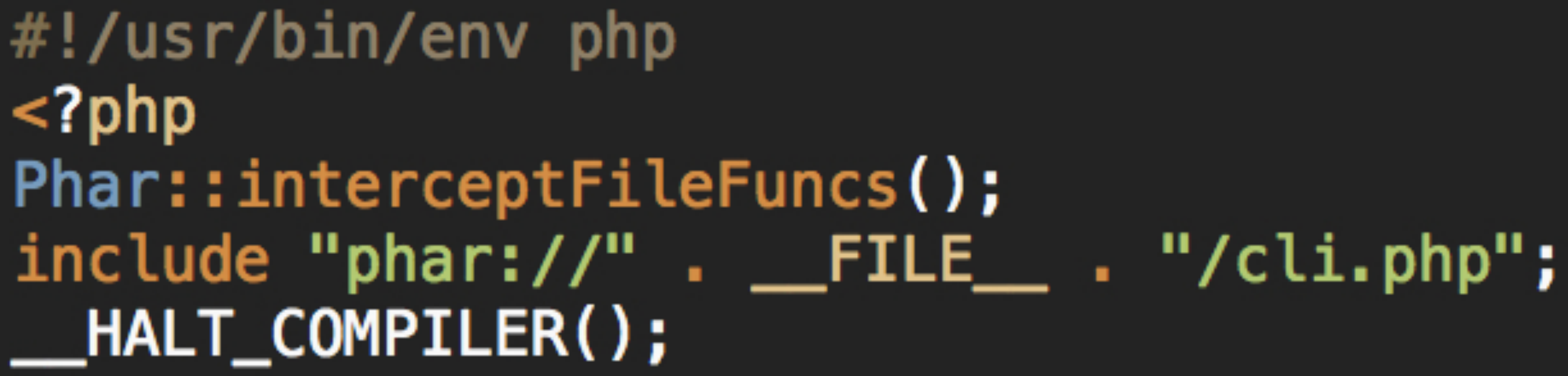
```php
Phar::interceptFileFuncs();
Phar::mungServer(array('REQUEST_URI', 'PHP_SELF', 'SCRIPT_NAME'));
Phar::WebPhar(null, 'web.php');
echo "Web Only version";
exit -1;
__HALT_COMPILER();
```

Phar::interceptFileFuncs()

Phar::mungServer()

Phar::WebPhar

CLI Phar



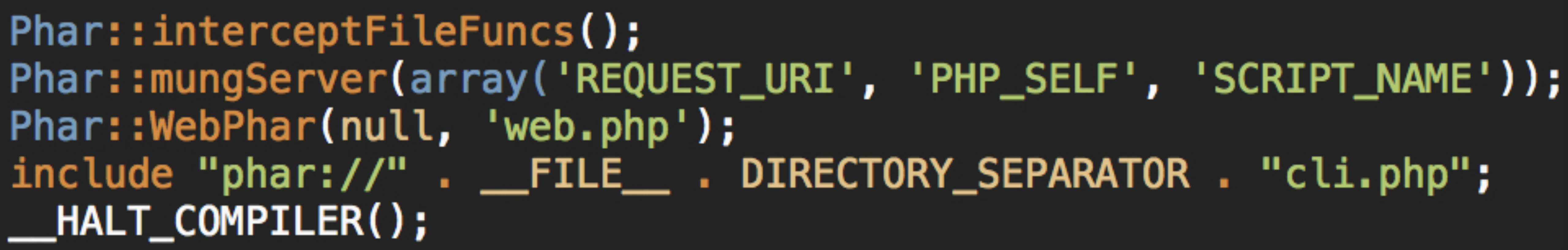
```php
#!/usr/bin/env php
<?php
Phar::interceptFileFuncs();
include "phar://" . __FILE__ . "/cli.php";
__HALT_COMPILER();
```

No Phar::WebPhar()

Simple file include

Shebang for easy execution

Web and CLI Phar



```php
Phar::interceptFileFuncs();
Phar::mungServer(array('REQUEST_URI', 'PHP_SELF', 'SCRIPT_NAME'));
Phar::WebPhar(null, 'web.php');
include "phar://" . __FILE__ . DIRECTORY_SEPARATOR . "cli.php";
__HALT_COMPILER();
```

CLI + Autoload

```
#!/usr/bin/env php
<?php
Phar::interceptFileFuncs();
function __autoload ($load) {
    include "phar://" . __FILE__ . "/$load.php";
}
include "phar://" . __FILE__ . "/cli.php";
__HALT_COMPILER();
```

CLI + Autoload

cli.php

```php
echo "This does CLI stuff\n";
$a = new a;
```

a.php

```php
class a {
    function __construct() {
        echo "This is class A\n";
        $b = new b;
    }
}
```

b.php

```php
class b {
    function __construct() {
        echo "This is class B\n";
    }
}
```

Phar offers a default stub

Will work for most

```
$phar = new Phar('project.phar');
$phar->setDefaultStub('cli.php', 'public/index.php');
echo $phar->getStub();
```

Extracts to temp dir if Phar is not available

Performance

Works with APC

Works on files <u>inside</u> the archive

Remember, everything goes via PHP

Extract static files to a directory and serve from there

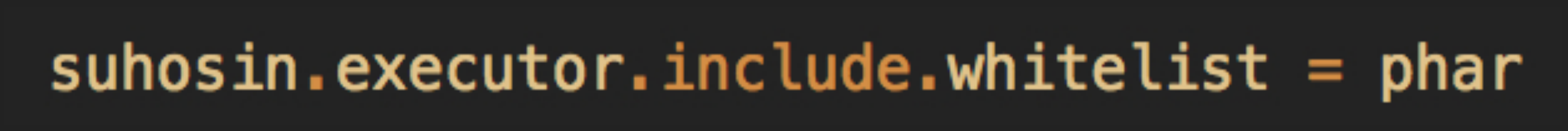
Security

Phar disallows writing by default

```
php -d phar.readonly=0 cli/create.php
```

Suhosin

Enabled by default in Debian and Ubuntu



```
suhosin.executor.include.whitelist = phar
```

Signatures are used to check the integrity of the archive

Signature

Hashes:

- MD5
- SHA1
- SHA256
- SHA512

OpenSSL

Hashes

By default executable Phar is signed SHA1

Tar / Zip are not unless specified

Extra Goodies!

Phar::mount()

Mount external files, such as config, into the phar archive

PharData

Phar archives can be extracted to disk

Can operate on non-phar gzip and tar

extract and compress archives

Like **PDO** for archives! :-)

Useful Tools

- phar-util
  - https://github.com/koto/phar-util
- empir
  - http://empir.sourceforge.net/

Get all the code used in this talk and more!

https://github.com/helgi/phar-talk

Come to the winter party tomorrow!
Beer and food is on me :-)

And some awesome swag!

Questions?

@h

helgi@engineyard.com

https://github.com/helgi/phar-talk

Joind.in: http://joind.in/4755