PHAR

Running applications out of the box
Costa Rica Tech Summit

Pablo Viquez
Senior PHP Developer, Open Source Platforms
pviquez@schematic.com

schematic

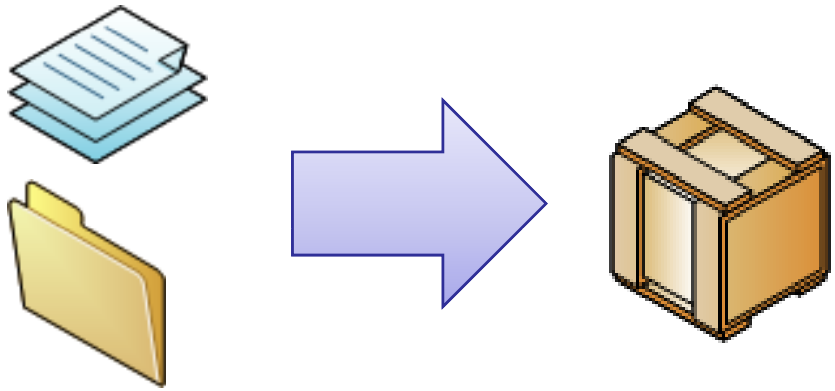
Agenda

- What are they
- Reasons to use them
- Requirements
- Installation
- How do they work
- PHAR Extension
- PHAR How to’s
- PEAR
- Current Issues
- Performance

What are they?

What are they?

- PHP->PHAR = Java.JAR
- A way to put entire PHP applications into a single file



- Single file that contains multiple files
- Can be accessed if it they were a regular files with filesystem functions

Why use it?

- Easy deployment.
- Security
  - Source inaccessible.
  - Files cannot be modify manually.
  - Write to file can be controlled.
- Portability
  - Very useful for libraries or code repositories.
- Compression
  - Zip or bz2

Requirements

Available starting PHP 5.2

Included in PHP 5.3

Compress Phar files:

- bzip2
- Zip

Signature verification:

- Hash

OpenSSL Signing :

- OpenSSL Extension

General usage (php extension)

- SPL

Installation

For simple usage:

- Just run it!
- Enable php_phar extension

2 options for creation

- PEAR package:
  - PHP_Archive_Creator
  - `pear install PHP_Archive-alpha`
- PHAR extension from PECL
  - `pecl install phar`

```
*nix:
extension=phar.so

Windows:
extension=php_phar.dll
extension=php_bz2.php
extension=php_zip.php
```

How does it work

How do the access to a regular file happens

```
$file = file_get_contents('foo.txt');
```

PHP “translate” the code into something like this:

```
$file =
    file_get_contents(
        'file://path/to/file/foo.txt');
```

How does it work

Making a regular include as follow:

```
require_once 'phar://path/to/bar.phar/foo.txt';
```

Since phar was implemented as a stream wrapper, it supports stream-handling functions such:

```
fopen()
file_get_contents()
opendir()
readdir()
copy()
rename()
```

Closer look…

Doing reflection on the class:

```
class Phar extends RecursiveDirectoryIterator
          implements RecursiveIterator,
                     SeekableIterator,
                     Traversable,
                     Iterator,
                     Countable,
                     ArrayAccess
{
    // ...
}
```

Closer look…

A phar archive can be one of three possible formats:

- ZIP
- TAR

The third is a customized file format designed specifically for phar, bringing more efficiency and performance.

- PHAR file format offers additional features such as intrinsic file-verification signatures.

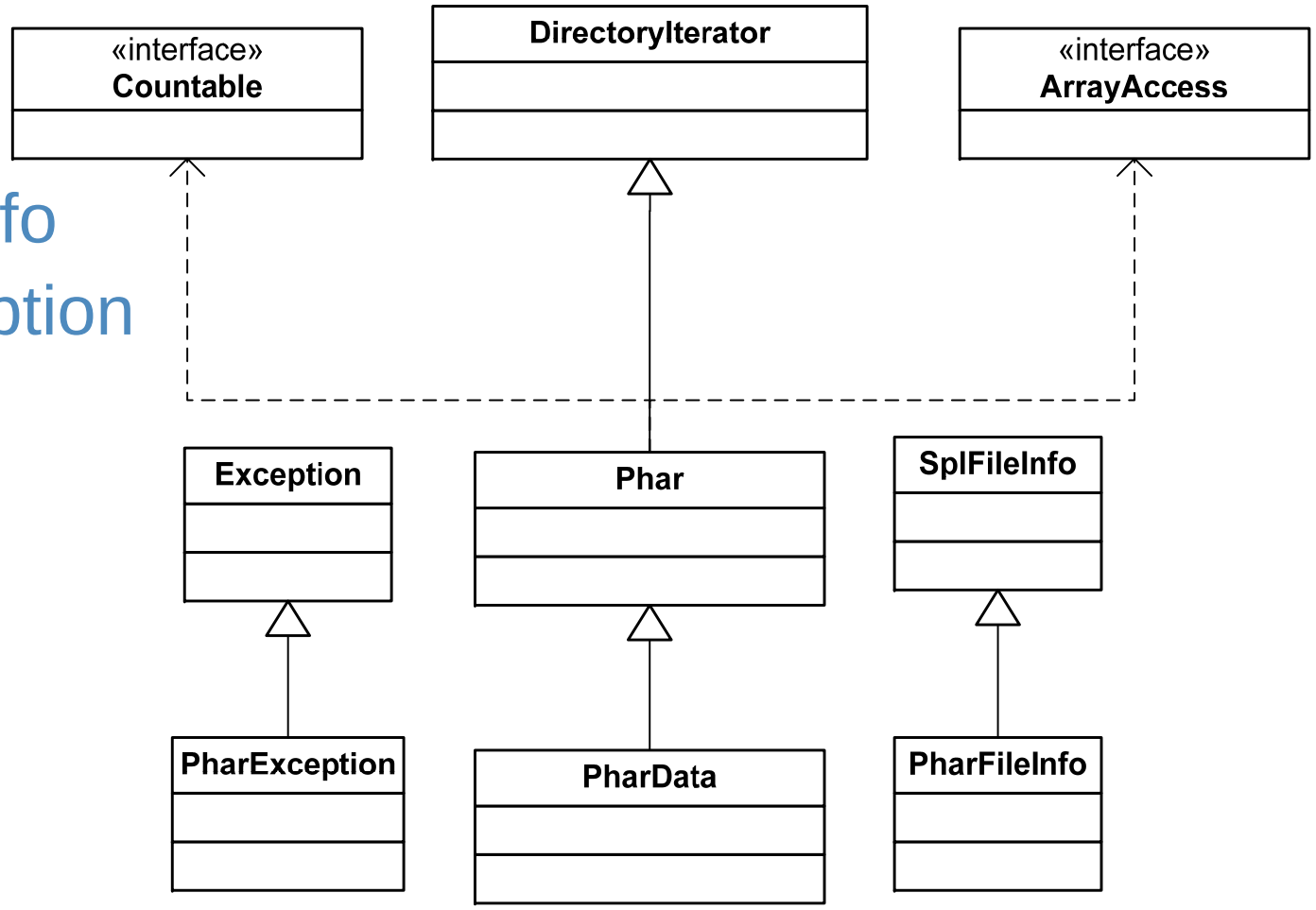
Closer look…

Featues	Phar	Tar	Zip
Stub file	Built-in	As file .phar/stub.php	As file .phar/stub.php
Can be execute without the extension?	Yes	No	No
Per file compression	zlib/bzip2	None	zlib (read/write) / bzip2 (read-only)
Whole archive compression	gz / bz2	gz / bz2	None
Archive signature	Yes	Yes	Not yet
Creation	Yes	Yes	Yes
Direct execution (php foo.phar)	Yes	Yes	Yes
Stream wrapper support	Yes	Yes	Yes
Works with phar object regardless of content	Yes	Yes	Yes
Can be created/mod with phar=readonly=1	No	Yes	Yes

Phar Extension Classes

The Phar extension makes available 4 classes:

- Phar
- PharData
- PharFileInfo
- PharException



schematic

Phar::Phar

- Supports reading and manipulation of Phar archives.

- Iteration through inherited functionality of the "RecursiveDirectoryIterator" class.

- Files inside a Phar can be accessed as an associative array, due the implementing of ArrayAccess interface.

- In order to be executable, a stub file is required.
  - A stub file, is used to set the PHP loader or bootstrap stub of a Phar archive
  - The stub file must end with the "_HALT_COMPILER()" token

Phar Ini File

```
[Phar]
phar.readonly = Off
phar.require_hash = Off
;phar.cache_list =
```

Phar

Phar: PHP Archive support	enabled
Phar EXT version	2.0.0-dev
Phar API version	1.1.1
CVS revision	$Revision: 1.370.2.44 $
Phar-based phar archives	enabled
Tar-based phar archives	enabled
ZIP-based phar archives	enabled
gzip compression	enabled
bzip2 compression	disabled (install pecl/bz2)
OpenSSL support	disabled (install ext/openssl)

Phar::Phar – How to create a Phar

```
void Phar::__construct (string $fname [,int $flags [, string $alias ]] )
```

```php
<?php
$p = new Phar('/path/foo.phar', 0, 'foo.phar');

// Start transaction
$p->startBuffering();

// add a new file via the array access API
$p['index.php'] = file_get_contents('/path/app/index.php');
$p['images/php.gif'] = file_get_contents('/path/app/images/php.gif');

$p->setMetaData(array('bootstrap' => 'index.php'));

// Save to disk
$p->stopBuffering();
```

Phar::Phar – Useful functions

Phar::webPhar()

Parses the $_SERVER['REQUEST_URI'] and simulates a web server, routing requests to the correct file.

Phar::webPhar() should only be called from the stub of a phar archive.

```
$phar = new Phar('/file/foo.phar', 0, 'foo.phar');
$phar->buildFromDirectory('/app/dir');
$phar->setStub('<?php
Phar::webPhar();
__HALT_COMPILER(); ?>');
```

```
void
    Phar::webPhar(
        string $alias,
        string $index,
        string $f404,
        array $mimetypes,
        array $rewrites)
```

Phar::Phar – Useful functions

Phar::interceptFileFuncs()

```
Void Phar::interceptFileFuncs()
```

Intercept fopen, file_get_contents, opendir, and all of the stat-related functions.

```
$a = new Phar('phpMyAdmin.phar.tar.php');
$a->startBuffering();
$a->setStub('<?php
Phar::interceptFileFuncs();
Phar::webPhar(
    "phpMyAdmin.phar",
    "phpMyAdmin-2.11.3-english/index.php");
__HALT_COMPILER();
');
$a->stopBuffering();
```

Phar::Phar – Useful functions

Phar::compress()

Compresses the entire Phar archive using Gzip or Bzip2 compression.

```
$p = new Phar('/file/foo.phar', 0, 'foo.phar');
$phar->buildFromDirectory('/app/dir');
$p1 =
    $p->compress(
        Phar::GZ); // copies to /path/to/my.phar.gz
$p2 =
    $p->compress(
        Phar::BZ2); // copies to /path/to/my.phar.bz2
```

```
object
    Phar::compress(
        int $compression
        [,string $extension])
```

Phar::Phar – Useful functions

Phar::buildFromIterator()

Construct a phar archive from an iterator..

```
$files =
    $phar->buildFromIterator(
        new RegexIterator(
            new DirectoryIterator('.'),
            '/^frontcontroller\d{0,2}\
.phar\.phpt\z|^\.\z|^\.\.\z/'),
    dirname(__FILE__) . DIRECTORY_SEPARATOR);
```

Directory iterator grabbing all files named frontcontroller*.phar.phpt and putting them into the phar archive.

```
array
    Phar::buildFromIterator(
        Iterator $iter
        [,string $base_directory])
```


schematic

PEAR - PHP_Archive

Phars created using the PHP_Archive package can executed using plain PHP, without any external dependencies at all.

PECL extension may require pecl/Phar to be enabled before they can be read, depending on the Phar version used to make them.

To create Phar files using the PEAR class, you need to use the PHP_Archive_Creator class

PEAR - PHP_Archive

PHP_Archive_Creator

- Can pre-process files before archiving them.
- Can strip unnecessary whitespace and comments.
- Alter require_once statements to refer to files within the archive.
- Create archives that use pecl/Phar if the extension happens to be available.
- Enables you to crawl a filesystem and recursively add files to an archive with a single method call.
- Sophisticated levels of wildcard-based file inclusion or exclusion within that filesystem.
- Does not create signatures by default
  - php.ini - `phar.require_hash = Off`

PEAR - PHP_Archive

```
// Use Phar_Archive_Creator from PEAR
require_once 'PHP/Archive/Creator.php';
$creator =
    new PHP_Archive_Creator(
        'index.php',
        'foo.phar');

$creator->addString(
    "<?php echo 'Hello World';",
    'index.php');

$creator->savePhar(
    dirname(__FILE__) . '/foo.phar');
```

PEAR - PHP_Archive

There are four PHP_Archive_Creator methods for adding files into your phar archive:

addString(string $contents, string $filename) - Adds a string as the contents of a file with the name $filename.

addArray(array $contents) - Adds an array of files to the phar archive.

addFile(string $file, string $filename) - similarly opens a file and adds it to the phar archive with the name $filename.

addDir(string $dir, array $ignore, array $include) - Traverses a directory and adds its files recursively to the phar archive.

- $ignore and $include - Arrays containing full paths to files or filesystem wildcards like *.php or i?m.txt.

PEAR - PHP_Archive

The biggest problem is handling calls that include external code.

Calls to __FILE__ or __autoload() offers no bigger problem.

Use of relative paths must be redefined:

```
// From:
require_once 'Foo/Bar.php';
// To:
require_once 'phar://foo.phar/Foo/Bar.php';
```

PEAR - PHP_Archive

PHP_Archive_Creator solves this issue by providing callbacks that replaces the calls for inclusion within the application:

simpleMagicRequire() - Uses calls to str_replace() to replace all instances of require_once and include_once with an internal phar require_once / include_once.

limitedSmartMagicRequire() - Uses preg_replace() to similarly replace instances of include or require that use quotations.

smartMagicRequire() - Uses preg_replace(), but replaces all instances of include and require.

tokenMagicRequire() - uses the tokenizer extension to scan PHP files token by token. Replaces any executable include/require statements, ignoring all uses of the words “include” or “require” in comments or variable names.

Current Phar Issues

As it’s now, packing an application into a Phar file could be quite challenging.

- The concept of "current directory" has no meaning in a stream wrapper.

- chdir() and getcwd() only work for directories stored on physical disks. All references to chdir() or getcwd() therefore needed to be modified by hand.

- All uses of opendir() that reference internal application code had to either use dirname(__FILE__) or be hand-modified.

Phar Performance

According to Phar Lead developer Greg Beaver, testing phpMyAdmin:

- Without APC: currently ~1.7x slower
- With APC: currently ~.3x slower

Thank You

Reference links:

- http://pecl.php.net/package/phar
- http://pear.php.net/package/pearweb_phars/download
- http://pecl4win.php.net/ext.php/php_phar.dll
- http://www.php.net/manual/en/intro.phar.php

We’re Hiring!

schematic