

PharScape

Introducing PHAR

Marcus Börger

Zend PHP conference 2008



Introducing Phar

- ✓ Archive format like JAR, TAR, ZIP
- ✓ Extension is not really required
- ✓ Normal PHP scripts that can be executed with PHP
- ✓ Command line tool which is itself a PHAR
- ✓ Deployable and executable format



Phar

- ✓ Entries are accessible as streams, phar://
- ✓ PHAR stub can contain PEAR package PHP_Archive
- ✓ Archive can be given an alias name
- ✓ Entries can be referenced by archive alias
- ✓ Entries can be compressed
- ✓ Extracted archives can still be accessed as phar://
- ✓ No need to change extracted files in any way
- ✓ Global and entry wide metadata supported



How to get PHAR



Using PEAR

```
$> pear install pecl/phar
```



Dependencies

- Extensions: SPL

- Optional exts: BZip2, OpenSSL, Zip

- Optional PECL exts: APC,



How to get PHAR

☑ Using PEAR

```
$> pear install pecl/phar
```

☑ Building your own from CVS

```
$> cvs -d  
:pserver:cvsread@cvs.php.net:/repository login  
phpfi
```

```
$> cvs -d  
:pserver:cvsread@cvs.php.net:/repository co -d  
phar pecl/phar
```

```
$> cd phar
```

```
$> phpize && ./configure && make
```

```
$> sudo make install
```

```
$> vi /etc/php.ini
```

```
$> make phar.phar
```

```
$> sudo cp phar.phar /usr/bin
```



How to get PHAR

☑ Using PEAR

```
$> pear install pecl/phar
```

☑ Building your own from CVS

```
$> wget http://pecl.php.net/get/phar-1.2.0.tgz
```

```
$> tar -xzvf phar-1.2.0.tgz
```

```
$> cd phar
```

```
$> phpize && ./configure && make
```

```
$> sudo make install
```

```
$> vi /etc/php.ini
```

```
$> make phar.phar
```

```
$> sudo cp phar.phar /usr/bin
```



How to get PHAR

- ☑ Default extension starting with PHP 5.3



Phar

- ☑ Packing files in a directory, using copy()

```
<?php
$phar = new Phar($argv[1], 0, 'newphar');
$dir  = new RecursiveDirectoryIterator($argv[2]);
$dir  = new RecursiveIteratorIterator($dir);

foreach($dir as $file)
{
    echo $file . "\n";
    copy($file, 'phar://newphar/' . $file);
}

?>
```


Phar

- ☑ Packing files in a directory, just a bit more

```
<?php
$phar = new Phar($argv[1], 0, 'newphar');
$dir  = new RecursiveDirectoryIterator($argv[2]);
$dir  = new RecursiveIteratorIterator($dir);
$dir  = new RegexIterator($dir, '/' . $argv[3] . '/');
$phar->startBuffering();
foreach($dir as $file)
{
    echo $file . "\n";
    copy($file, 'phar://newphar/' . $file);
}
$phar->compressAllFilesBZIP2();
$phar->stopBuffering();
?>
```

Phar

- ☑ Packing files in a directory, using ArrayAccess

```
<?php
$phar = new Phar($argv[1], 0, 'newphar');
$dir = new RecursiveDirectoryIterator($argv[2]);
$dir = new RecursiveIteratorIterator($dir);
$dir = new RegexIterator($dir, '/' . $argv[3] . '/');
$phar->startBuffering();
foreach($dir as $file) {
    $f = $dir->getSubPathName();
    echo $f . "\n";
    $phar[$f] = file_get_contents($file);
}
$phar->compressAllFilesBZIP2();
$phar->stopBuffering();
?>
```

Self referencing PHARs

- ☑ PHAR stubs can reference themselves

```
<?php
Phar::mapPhar('myphar.phar', 'myphar');
include 'phar://myphar/main.php';
__HALT_COMPILER();
?>
```

```
$> php myphar.phar
```



Self referencing PHARs

- ☑ PHAR stubs might still be used when extracted

```
<?php
$1 = Phar::getExtractList();
if(!isset($1[___FILE___]) && !isset($1['phar://myphar'])) {

    Phar::mapPhar('myphar.phar', 'myphar');
}
include 'phar://myphar/main.php';
__HALT_COMPILER();
?>
```

```
$> php myphar.phar
```



Self referencing PHARs

- ☑ PHAR stubs can use metadata for bootstrapping

```
#!/usr/bin/php
<?php
$phar = new Phar(__FILE__);
$meta = $phar->getMetaData();
if (isset($meta['boot'])) {
    require_once 'phar://' . __FILE__ . '/' . $meta['boot'];
}
__HALT_COMPILER();
?>
```

```
$> phar.phar meta-set -f myphar.phar -k boot -m main.php
$> ./myphar.phar
```

Self referencing PHARs

- ☑ PHAR stubs can implement `__autoload`

```
#!/usr/bin/php
<?php
function __autoload($cn) {
    $f = 'phar://' . __FILE__ . '/' . strtolower($cn) . '.inc';
    if (file_exists($f)) {
        require($f);
    }
}
new MyPhar();
__HALT_COMPILER();
?>
```

```
$> phar.phar meta-set -f myphar.phar -k boot -m main.php
$> php myphar.phar
```



phar.phar

- ☑ Command line tool to deal with PHAR packages

```
$> phar.phar
```

```
No command given, check /usr/bin/phar.phar help
```

```
$> phar.phar help-list
```

```
compress extract help help-list info list meta-del  
meta-get meta-set pack sign stub-get stub-set tree
```

```
$> phar.phar help help
```

```
help This help or help for a selected command.
```

```
Optional arguments:
```

```
... Optional command to retrieve help for.
```



Deployment ideas

- ✓ Include and require are hard to maintain
- ✓ __autoload is slow



Deployment ideas

- ✓ Include and require are hard to maintain
- ✓ __autoload is slow
- ✓ During development and testing:

```
function __autoload($name) {  
    global $db;  
    require $name . '.inc';  
    $db->add($_SERVER['SCRIPT_FILENAME'], $name);  
}
```

Deployment ideas

- ✓ Include and require are hard to maintain
- ✓ `__autoload` is slow

- ✓ During development and testing:

```
function __autoload($name) {  
    global $db;  
    require $name . '.inc';  
    $db->add($_SERVER['SCRIPT_FILENAME'], $name);  
}
```

- ✓ During deployment:

- ✓ Prepend files with require statements
- ✓ Pack all into a single PHAR
- ✓ Use INI setting `phar.cache_list`



Phar::webPhar

- ☑ A webserver (application) in a single phar

```
$phar = new Phar('myphar.phar');  
$phar['index.php'] = '<?php echo "Hello World"; ?>';  
$phar['index.phps'] = '<?php echo "Hello World"; ?>';  
$phar->setStub('<?php  
Phar::webPhar();  
__HALT_COMPILER(); ?>');
```



PharData

- ☑ These archives cannot be executed (no stub)
- ☑ Can be Tar or Zip based
- ☑ Can be modified even if `phar.readonly=ON`



Hashing & Signing



Archives can be signed

MD5

SHA1 default for Phar and Tar based

SHA256

SHA512

OpenSSL requires OpenSSL extension



INI

- ☑ Prevent modification of archives in production
 - ☑ `phar.readonly=ON`

- ☑ Require signed (hashed) archives
 - ☑ `phar.require_hash=ON`

- ☑ Speed up by caching and pre-parsing
 - ☑ `phar.cache_list=<archives>...`



Known Issues

- ☑ Truncation of GZ/BZip2 archives prior to PHP 5.2.6
- ☑ INI setting `phar.extract_list` gone in Phar 2.0
- ☑ PEAR package `PHP_Archive` not fully compatible
 - ☑ Does not implement `ArrayAccess`
 - ☑ Does not implement `Iterator`



At Last some Hints

- ✓ List of all SPL classes PHP 5.0.0
`php -r 'print_r(array_keys(spl_classes()));'`
- ✓ Reflection of a built-in class PHP 5.1.2
`php --rc <Class>`
- ✓ Reflection of a function or method PHP 5.1.2
`php --rf <Function>`
- ✓ Reflection of a loaded extension PHP 5.1.2
`php --re <Extension>`
- ✓ Extension information/configuration PHP 5.2.2
`php --ri <Extension>`



THANK YOU

- ☑ This Presentation
 - <http://talks.somabo.de/20080917.pdf>
 - <http://talks.somabo.de/20080917.pps>
- ☑ Phar
 - <http://pecl.php.net/packages/phar>
 - <http://php.net/phar>
- ☑ SPL Documentation
 - <http://php.net/~helly>

