Plein PHAR !

Forum PHP 2010
9 et 10 novembre
Paris

cité des sciences et de l'industrie

Les scripts utilisés pour cette conférence sont disponibles à l'adresse

http://medias.mageekbox.net/conferences/forumPHP2010/

Frédéric Hardy

frederic.hardy@mageekbox.net \ http://blog.mageekbox.net \ @mageekguy

Plein PHAR !

Préambule

```
# php forumPHP2010.phar -e
```

suhosin.executor.include.whitelist="phar"

Plan

1. Feux de position
2. Feux de croisement
3. Feux de route

Feu de position

1. Description
2. Historique
3. Installation

Description

- PHAR = PHP Archive.
- PHAR est à PHP ce que JAR est à Java.
- Concatène plusieurs fichiers en un seul.
  - Facilite la distribution.
  - Permet l'exécution via un seul fichier
    - Via différentes SAPI.
- Supporte divers format de compression.

Historique

- Extension PECL
  - Créée par
    - Steph Fox (php|architect).
    - Greg Beaver (contributeur PEAR/PECL).
    - Marcus Börger (parmi les auteurs de PHP 5).
  - Version 1 en 2007
  - Version 2 en 2009
- Intégrée à PHP depuis PHP 5.3.

Configuration minimum

- PHP version 5.2 et supérieure.
- SPL, intégrée à PHP 5.3.
- Compression
  - Extension PHP zlib.
  - Extension PHP bzip2.
- Sécurisation
  - Extension openssl.

Directives

- Ne pas utiliser `--disable-phar` lors du `./configure`.
- Ne pas utiliser `--zend-enable-multibyte` lors du `./configure`.
- `phar.readonly`
- `phar.require_hash`
- `phar.cache_list`

Feu de croisement

1. Fonctionnement
2. Compatibilité
3. Structure
4. Création
5. Utilisation
6. Modification

Fonctionnement

- Rassemble plusieurs fichiers de nature différente en un seul.
- Lisible par PHP directement.
  - Flux `phar://`.
- Informations extraites à la demande.
- Utilisable par toutes les fonctions supportant les flux.

Compatibilité

- Au niveau de PHP :
  - Utilisable uniquement par PHP 5.3+ et l'extension PECL correspondante.
- Pour le reste du monde :
  - Les PHAR au format PHAR ne sont lisibles que par PHP.
  - Les PHAR au format ZIP/TAR sont lisibles par les utilitaires concernés.

Structure

1. Un fichier de démarrage (stub).
2. Un manifeste décrivant la structure du PHAR.
   - Chemin d'accès + permissions
3. Les fichiers.
4. Une signature optionnelle.

Le fichier de démarrage

- Interprété :
  - lorsque le PHAR est exécuté.
  - Lorsque le PHAR est inclus dans un script PHP.
- Permet de définir l'environnement d'exécution du PHAR :
  - Autoload.
  - Lecture de fichiers de configuration externe.
  - etc.

Création

- Création impossible par défaut :
  - `phar.readonly` à 0 dans le php.ini.
- `-d phar.readonly = 0` en ligne de commande.
  - À ne pas utiliser en production.

Plein PHAR !

Script

```
<?php

$phar = new \Phar('helloForumPHP2010.phar');
$phar['index.php'] =
   '<?php echo "Hello Forum PHP 2010 !\n"; ?>'
;
$phar->createDefaultStub();

?>
```

```
 # php -d phar.readonly 001-helloForumPHP2010.php
 # php helloForumPHP2010.phar
```

Définition du fichier de démarrage

- Doit :
  - contenir du code PHP valide.
  - se terminer par `__HALT_COMPILER();`.
- La balise `?>` n'est pas obligatoire.
- Peut être extrait à l'aide de `\Phar::getStub()`.

```
<?php

$phar = new \Phar('helloworld.phar');
var_dump($phar->getStub());

?>
```

Les fichiers de démarrage par défaut

- Intégrable via :
  - `\Phar::createDefaultStub()`
  - `\Phar::setDefaultStub()` et `\Phar::setStub()`.
- Utilisable en ligne de commande et sur le Web.

```
<?php
$phar = new \Phar('helloForumPHP2010.phar');
$phar['cli.php'] = '<?php echo "Hello Forum PHP 2010 !\n"; ?>';
$phar['web.php'] =
    '<?php echo "<h1>Hello Forum PHP 2010 !</h1>"; ?>';
$phar->setDefaultStub('cli.php', 'web.php');
# identique à $phar->setStub($phar->createDefaultStub(
#          'cli.php', 'web/index.php'
#      )
# );
?>
```

Ajout de fichiers

- Plusieurs méthodes possibles
  - Via l'interface `\arrayAccess`.
  - Via `\Phar::addFile()`.
  - Via `\Phar::addFromString()`.
  - Via `\Phar::addEmptyDir()`.
  - Via `\Phar::buildFromDirectory()`.
  - Via `\Phar::buildFromIterator()`
    - Doit retourner des instances de `\SplFileInfo`.

Signature de l'archive

- Permet de vérifier l'intégrité d'un PHAR.
- `\Phar::setSignatureAlgorithm()`, accepte `\Phar::MD5`, `\Phar::SHA1`, `\Phar::SHA256`, `\Phar::SHA512` et `\Phar::OPENSSL`.
- `\Phar::getSignature()` permet d'extraire d'une archive PHAR sa signature.

```
<?php

$phar = new \Phar('helloForumPHP2010.phar');
$phar->setSignatureAlgorithm(\phar::SHA1);
$phar->setStub('<?php $phar = new \phar(__FILE__);
    var_dump($pharData->getSignature());
    include(\'phar://helloForumPHP2010.phar/index.php');
    __HALT_COMPILER();'
);

?>
```

Utilisation

- En ligne de commande.
- Via `include`/`require`.
- Via le flux `phar://` et les fonctions les supportants.
- Via `\recursiveDirectoryIterator`.
- Via les interfaces :
  - `\arrayAccess`.
  - `\countable`.
  - `\iterator`.
  - `\traversable`.

Exemples

```
<?php

include('phar://helloForumPHP2010.phar/index.php');

file_get_contents('phar://helloForumPHP2010.phar/index.php');

$phar = new \phar('helloForumPHP2010.phar');

var_dump($phar['index.php']);
var_dump($phar['cli.php']);
var_dump($phar['web.php']);

foreach ($phar as $file) {
    var_dump($file);
}

var_dump(sizeof($phar);

?>
```

Extraction de fichiers

- Possible à l'aide de `\Phar::extractTo()`.
- Permet d'extraire la totalité ou seulement certains fichiers d'une archive.

```
<?php

$phar = new \phar('helloForumPHP2010.phar');
$phar->extractTo(__DIR__, array('index.php'), true);

?>
```

Feux de route

1. Méta-données
2. Formats et Compressions
3. Point de montage.
4. Tests unitaires
5. Performances
6. Cas pratique

Méta-données

- `\Phar::(set|has|get|del)MetaData()`

```
<?php

$phar = new \Phar('helloForumPHP2010.phar');
$phar->setMetaData(array(
    'country' => 'France',
    'city' => 'Paris',
    'location' => 'Cité des sciences & de l\'industrie',
    'creator' => 'AFUP',
    'date' => date('d/m/Y h:i:s')
));
$phar->setStub('<?php $phar = new \phar(__FILE__);
    var_dump($pharData->getMetaData());
    include(\'phar://helloForumPHP2010.phar/index.php');
    __HALT_COMPILER();'
);

?>
```

Formats et compression

- PHAR non exécutable et exécutable
- Formats disponibles :
  - `\Phar::ZIP`.
  - `\Phar::TAR`.

Compressions possibles :
`\Phar::NONE`.
`\Phar::GZ` (zlib).
`\Phar::BZ2` (bzip2).

	\Phar::TAR	**\Phar::ZIP**
\Phar::NONE	.tar	.zip
\Phar::GZ	.tar.gz	.zip
\Phar::BZ2	.tar.bz2	.zip

PHAR non exécutable

- Le nom de l'archive ne doit pas contenir `.phar`.
- N'a pas de fichier de démarrage.
- Via `\PharData::__construct()` ou `\Phar::convertToData()`.

```
<?php

$phar = new \Phar('helloForumPHP2010.phar');
$pharData = $phar->convertToData(\Phar::TAR);
var_dump($pharData->getStub();
$pharData = $phar->convertToData(\Phar::TAR, \Phar::GZ);
var_dump($pharData->getStub();

?>
```

PHAR exécutable

- Le nom de l'archive doit contenir `.phar`.
- Création :
  - Via `\Phar::__construct()` et `\Phar::setStub()`.
  - Via `\PharData::convertToExecutable()`
    - Attention au fichier de démarrage.
- Via `\Phar::convertToExecutable()`.

```
<?php

$pharData = new \PharData('helloForumPHP2010.tar');
$phar->convertToExecutable(\Phar::PHAR, \Phar::GZ);
$phar->createDefaultStub('cli.php', 'web.php');

?>
```

Compression locale et/ou globale

- Possible de ne compresser que certain fichier dans un PHAR :
  - `\Phar::compressFiles()`.
  - `\PharData::compressFiles()`.
- Possible de compresser globalement un PHAR :
  - `\Phar::compress()`.
  - `\PharData::compress()`.

Décompression

- Implicite lors d'une utilisation.
- Possible explicitement à l'aide des méthodes :
  - `\Phar::decompress()`.
  - `\PharData::decompress()`.
  - `\Phar::decompressFiles()`.
  - `\PharData::decompressFiles()`.

Point de montage

- `\Phar::mount()` permet d'intégrer un fichier externe à un PHAR.
- Utile pour la configuration, les journaux d'événements, etc.
- Le montage peut se faire de l'intérieur ou de l'extérieur de l'archive.

```
<?php

$pharData = new \PharData('helloForumPHP2010.tar');
$phar->setStub('<?php
    \Phar::mount(__DIR__ . "/config.php", "config.php");
    require("config.php");
    echo "Hello $who !\n";
    __HALT_COMPILER();'
);

?>
```

Tests unitaires

- Comment tester la création d'un PHAR ?
  - Validation du fichier de démarrage ?
  - Vérification des fichiers inclus ?
  - Signature du bon type ?
  - etc.

Tests unitaires

- Solution ?
  - Créer effectivement le PHAR.
    - Et si environnement de développement avec PHP < 5.3 ?
  - Injection de dépendances
  - Stub/Mock

Plein PHAR !

Exemple

```
$this->assert

    ->object($generator->run())->isIdenticalTo($generator)

    ->mock($phar)->call('__construct', array(

            $generator->getDestinationDirectory() . '/' .
              atoum\phar\generator::phar, null, null

            ))

    ->call('setMetadata', array(array(

                'version' => atoum\test::getVersion(),
                'author' => atoum\test::author,
                ...

            )))

    ->call('setStub', array('<?php \phar::mapPhar(\'' . phar\generator::phar . '\');
       require(\'phar://' . phar\generator::phar . '/classes/autoloader.php\'); if
       (PHP_SAPI === \'cli\')...

->call('setSignatureAlgorithm', array(\phar::SHA1,null)

...
```

Performances

- `\Phar::TAR` plus efficace que `Phar::ZIP`
  - Mais noms de fichiers limité à 255 caractères.
- Perte de performance légère sans compression.
- Perte de 10 à 15% avec compression.
- L'utilisation de la directive `phar.cache_list` règle le problème.
- APC peut augmenter les performances jusqu'à un facteur 6 !

Plein PHAR !

Une preuve ?

```
# php scripts/runners/directory.php -d tests/units/classes/
> Atoum version 133 par Frédéric Hardy.
...
> Total tests duration: 3.76 seconds.
> Total tests memory usage: 47.25 Mb.
> Running duration: 9.91 seconds.
> Success (56 tests, 228 methods, 3615 assertions) !
```

```
# php mageekguy.atoum.phar -testIt
> Atoum version 133 par Frédéric Hardy.
...
> Total tests duration: 2.18 seconds.
> Total tests memory usage: 36.50 Mb.
> Running duration: 6.34 seconds.
> Success (56 tests, 228 methods, 3615 assertions) !
```

Un cas pratique : Atoum

- En ligne de commande

```
# php mageekguy.atoum.phar --help

Usage: php mageekguy.atoum.phar [options]

Atoum version 105 by Frédéric Hardy.

Available options are:

                    -h, --help: Display this help

                  -i, --infos: Display informations

              -s, --signature: Display phar signature

   -e <dir>, --extract <dir>: Extract all file in <dir>
```

Atoum

- Dans le code

```
<?php

require 'mageekguy.atoum.phar';

class boolean extends \mageekguy\atoum\test {
    public function testTrue() {
        $this->assert->boolean(true)->isTrue();
    }
}

?>
```

Références

- http://php.net/phar
- http://blog.pascal-martin.fr/post/php-5.3-phar-php-archive

Plein PHAR !

Questions ?

Remerciements

- Merci à l'AFUP.
- Merci à PMSIpilot.
- Merci à vous.

Cette conférence est maintenant terminée, vous pouvez reprendre une activité normale.

frederic.hardy@mageekbox.net \ http://blog.mageekbox.net \ @mageekguy