PHP : Gestion de paquets

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

Autochargement (alias autoloading)

Se charge de l'inclusion automatique de fichiers

- Evite d'expliciter tous les fichiers à inclure.
- Utilisé pour l'usage d'extensions/librairies tierces.
- Mis en oeuvre par appel de `spl_autoload_register` avec une fonction de chargement (`spl_autoload` par défaut) :
  - Cette dernière sera appelée en cas d'utilisation d'une classe/fonction `X` non déclarée.
  - Et tentera d'inclure un fichier système sur la base du nom de `X`.
- Repose sur la mise en correspondance d'espaces de noms (alias namespaces) et de répertoires système.

Exemple d'autochargement

CamelCase ↔ ./camel_case.php (spl-autoload1.php)

```php
spl_autoload_register(function ($class_name) {
 $str = strtolower(preg_replace('/(?<!^)[A-Z]/','_$0',$class_name));
 include $str . '.php';
});
$obj  = new UneClasse(); // déclenche : include "une_classe.php";
// erreur : fichier introuvable
$obj2 = new PasDeClasse(); // déclenche : include "pas_de_classe.php";
```

Espace de noms (alias namespaces)

Le pendant en programmation de la notion de répertoire

Identifie un groupe de classes, interfaces, fonctions et constantes

- Evite les collisions entre noms des éléments (classes, ...) créés ou importés.
- Autorise les alias pour dénommer espaces et sous-espaces de noms.

Espace de noms : déclaration et résolution

Se déclare avec mot-clé `namespace`

On peut imbriquer les espaces de noms avec `\`

Différentes règles de résolution s'appliquent

`$obj = new Foo();` (*unqualified*)

- `currentnamespace\Foo` ou `Foo`

`$obj = new subnamespace\Foo();` (*qualified*)

- `currentnamespace\subnamespace\Foo` ou `subnamespace\Foo`

`$obj = new \currentnamespace\Foo();` (*fully-qualified*)

- `currentnamespace\Foo`

Espace de noms : aliasing

namespaces.php

```php
namespace foo;
use My\Full\Classname as Another;
// this is the same as use My\Full\NSname as NSname
use My\Full\NSname;
// importing a global class
use ArrayObject;
// importing a function
use function My\Full\functionName;
// aliasing a function
use function My\Full\functionName as func;
// importing a constant
use const My\Full\CONSTANT;
$obj = new namespace\Another; // instantiates object of class foo\Another
$obj = new Another; // instantiates object of class My\Full\Classname
NSname\subns\func(); // calls function My\Full\NSname\subns\func
$a = new ArrayObject(array(1)); // instantiates object of class ArrayObject
// without the "use ArrayObject" we would instantiate an object of class foo\ArrayObject
func(); // calls function My\Full\functionName
echo CONSTANT; // echoes the value of My\Full\CONSTANT
```

Gestionnaires de paquets PHP

Composer, PEAR, PECL

- Systèmes de distribution de paquets PHP (archives PHP `.phar`) avec gestion automatique des dépendances.
- Sites de diffusion associés.

Composer

- Place les paquets dans un répertoire choisi du projet (eg. `vendor`).
- Stocke les méta-données d'installation (noms et versions de paquets, . . .) dans deux fichiers JSON qui sont à engager dans le VCS.
- Si nécessaire, génère un autoloader PSR-4 pour importer les fonctionnalités de paquets par espaces de noms.
- Site de diffusion : Packagist.

Composer : installation et usage

Exemple d'installation par projet (sans vérification SHA ! !). Se loger sur Janus.

```
$ cd Mes_projets_web
$ mkdir Composer
$ cd Composer
$ php -r "copy('https://getcomposer.org/installer',
'composer-setup.php');"
$ php composer-setup.php
-install-dir=$HOME/Mes_projets_web/Composer
$ php -r "unlink('composer-setup.php');"
```

Liste des options

```
$ php composer.phar
```

Composer : exemple d'installation de paquet

Créer `composer.json`

```
1 {
2   "require": {
3       "monolog/monolog": "^1.22"
4    }
5 }
```

Installer

```
$ php composer.phar install
// crée le répertoire vendor
// y installe le(s) paquet(s) - ici monolog -
// et autoloader
```

Utiliser le paquet `monolog` dans un script (composer-monolog.php)

```
1 // autoloader créé et mis à jour par Composer
2 require __DIR__.'/vendor/autoload.php';
3 // usage des espaces de noms de Monolog
4 use Monolog\Logger;
5 use Monolog\Handler\StreamHandler;
6 // usage des fonctionnalités de Monolog
```

PEAR et PECL

PEAR (PHP Extension and Application Repository)

- Une librairie structurée de code source libre pour les utilisateurs de PHP.
- Un système de distribution du code source et de maintenance des paquets.
- Un style de codage pour les programmes écrits en PHP.
- Un site Web, des listes de diffusion et des sites miroirs pour supporter la communauté PHP/PEAR.

PECL (PHP Extension Community Library)

- Projet distinct de PEAR pour distribuer les extensions de PHP (code écrit en C et compilé telle que l'extension PDO).
- Les extensions PECL sont aussi distribuées en paquets et peuvent être installées avec l'installeur de PEAR.