SIN6U5 : Développement web 2 (Prog. web coté serveur) Espaces de noms et le autoloader de composer

Luigi Santocanale
LIF, Aix-Marseille Université

29 mars 2018

Plan

Plan

Objectfs des espaces de noms

- Regrouper des classes, interfaces, fonctions ou constantes.
- Résoudre les collisions de noms entre
  - votre code que vous créez ;
  - les classes, fonctions ou constantes internes de PHP ;
  - celles de bibliothèques tierces.
- Améliorer la lisibilité du code par des alias ou des raccourcis. Exemple

$$\texttt{Noms\_Extrememement\_Long} \quad \mapsto \quad \texttt{NEL}$$

Cf. :

- le système des fichiers dans un système d'exploitation ;
- les packages en `java` ;
- les modules en `python`.

Définition d'un espace de noms : `namespace`

```php
<?php
// testNamespace.php

namespace Luigi\MonProjet ;

interface iTest {
    public static function test();
}

class Test implements iTest {
    public static function test() { echo \strlen("test1")."\n" ;}
}

const TEST = "test2\n";

function test(){
    echo TEST ;
}
```

- `namespace` définit l'espace de noms courant ;
- tout (classes, interfaces, fonctions, constantes) ce qui est défini dans ce fichier appartient à (est encapsulé dans) cet espace ;
- tout ce qui est utilisé dans ce fichier appartient à cet espace ;
- on accède à l'*espace global* en préfixant par le symbole \ (antislash) comme pour `\strlen`.

Accès à d'autres espace(s) de noms

```php
<?php

// testNamespace.php

namespace Luigi\MonProjet ;

interface iTest {
    public static function test();
}

class Test implements iTest {
    public static function test() {
 ↪    echo \strlen("test1")."\n" ;}
}

const TEST = "test2\n";

function test(){
    echo TEST ;
}
```

```php
<?php
namespace Luigi;

include 'testNamespace.php';

MonProjet\Test::test();
\Luigi\MonProjet\Test::test();
MonProjet\test();
\Luigi\MonProjet\test();
```

Notez l'utilisation des chemins relatifs/absolus par rapport à l'espace de noms courant.

Remarques

- La déclaration (mot clés `namespace`) doit se trouver au début du fichier.
- Si pas de déclaration d'espace de noms,
  alors l'espace de noms global—la racine, denotée par le symbole \ —est sous-entendu.
- La portée d'une déclaration est locale au fichier :

```php
<?php

// testScope_included.php

function hello(){
    echo "From included file\n";
}
```

```php
<?php

// testScope_including.php

namespace A;

include 'testScope_included.php';

function hello(){
    echo "From including file\n";
}

hello();
\hello();
```

Remarques

- Les fonctions et classes natives PHP se trouvent dans l'espace de noms global.
- L'utilisation de l'antislash pour l'espace de noms global est
  - **obligatoire**, pour les classes et les interfaces,
  - *optionnelle*, pour les fonctions et les constantes.
- Dans le deuxième cas, les noms sont résolus
  - d'abord par rapport à l'espace de noms courant,
  - puis, par rapport à la racine.

Aliases : use

```php
<?php
namespace Luigi;

include 'testNamespace.php';

use Luigi\MonProjet\Test; // alias d'une classe
use Luigi\MonProjet as MP ; // alias d'un espace de noms
use function Luigi\MonProjet\test as test2 ; // alias d'une fonction
use const Luigi\MonProjet\TEST as MYCONST ; // alias d'une constante

Test::test();
MP\Test::test();
test2();
echo MYCONST ;
```

Caveats

Attention :

- l'espace de noms avec `use` est toujours absolu.

Le code suivant :

```php
<?php
namespace Luigi;
include 'testNamespace.php';
use MonProjet\Test as Test;
Test::test();
```

ne marche pas.

Plan

Mon autoloader

```php
<?php
function autoload($className) {
    global $config;
    $fileName = "$className.php";
    $pathToMyClasses = $config['base'].'/src/classes/';
    $paths = array('./', $pathToMyClasses);

    $found = false;
    foreach ($paths as $path) {
        if (file_exists("$path/$fileName")) {
            require_once "$path/$fileName";
            $found = true;
        } else {
            $directories = scandir($path);
            foreach ($directories as $directory) {
                if (is_dir("$path/$directory") &&
                        file_exists("$path/$directory/$fileName")) {
                    require_once "$path/$directory/$fileName";
                    $found = true;
                    break;
                }
            }
        }
        if ($found) {
            break;
        }
    }
}

spl_autoload_register('autoload');
```

Plan

Le fichier `composer.json`

Décrit la structure d'un projet/paquet, incluant :

- les dépendances d'autres paquets,
- des définitions d'espaces de noms.

```
{
    "name": "luigi/test",
    "type": "library","version":"0.0.0",
    "description": "Exemples utilisation composer et son autoload",
    "license": "proprietary",
    "authors": [
        {"name": "Luigi Santocanale", "email":
         "luigi.santocanale@lis-lab.fr"}],
    "require": {
        "twig/twig": "^2.4",
        "symfony/validator":          "2.1.*",
        "doctrine/dbal":              "2.2.*",
        "monolog/monolog":            "dev-master"
    },
    "autoload": {
        "psr-4": {
            "MesClasses\\":         ["src/","lib/"]
        }
    }
}
```

Installation des paquets

```
composer install
```

ou

```
composer update
```

Répertoires créés

```
.
`-- vendor
    |-- composer
    |-- doctrine
    |   |-- common
    |   |   |-- lib
    |   |   `-- tests
    |   `-- dbal
    |       |-- bin
    |       |-- lib
    |       `-- tests
    |-- monolog
    |   `-- monolog
    |       |-- doc
    |       |-- src
    |       `-- tests
    |-- psr
    |   `-- log
    |       `-- Psr
    |-- symfony
    |   |-- polyfill-mbstring
    |   |   `-- Resources
    |   `-- validator
    |       `-- Symfony
    `-- twig
        `-- twig
            |-- doc
            |-- lib
            |-- src
            `-- test
```

`composer.json` dans un projet

Chaque projet mappe son propre `namespace` vers son code. Par exemple :

```
{
    "name": "doctrine/dbal",
    "type": "library","version":"2.2.2",
    "description": "Database Abstraction Layer",
    "keywords": ["dbal", "database", "persistence", "queryobject"],
    "homepage": "http://www.doctrine-project.org",
    "license": "LGPL",
    "authors": [
        {"name": "Guilherme Blanco", "email":
↪    "guilhermeblanco@gmail.com"},
        {"name": "Roman Borschel", "email": "roman@code-factory.org"},
        {"name": "Benjamin Eberlei", "email": "kontakt@beberlei.de"},
        {"name": "Jonathan Wage", "email": "jonwage@gmail.com"}
    ],
    "require": {
        "php": ">=5.3.2",
        "doctrine/common": ">=2.2.0,<=2.2.99"
    },
    "autoload": {
        "psr-0": { "Doctrine\\DBAL": "lib/" }
    }
}
```

Contenu du répertoire `vendor/composer`

```
-rw-r--r--  1 lsantoca  staff  13248 29 mar 17:20 ClassLoader.php
-rw-r--r--  1 lsantoca  staff   1075 29 mar 17:20 LICENSE
-rw-r--r--  1 lsantoca  staff    147 29 mar 17:20 autoload_classmap.php
-rw-r--r--  1 lsantoca  staff    243 29 mar 17:20 autoload_files.php
-rw-r--r--  1 lsantoca  staff    419 29 mar 17:20 autoload_namespaces.php
-rw-r--r--  1 lsantoca  staff    484 29 mar 17:20 autoload_psr4.php
-rw-r--r--  1 lsantoca  staff   2414 29 mar 17:20 autoload_real.php
-rw-r--r--  1 lsantoca  staff   2552 29 mar 17:20 autoload_static.php
-rw-r--r--  1 lsantoca  staff  14402 29 mar 17:20 installed.json
```

Inclusion de l'autoloader

La première lingne fde votre `index.php` rassemblera à ca :

```php
<?php
$rootDir = __DIR__ . '/..';
require_once "$rootDir/vendor/autoload.php";
```

Le fichier `vendor/autoload.php` n'est qu'une sorte de alias :

```php
<?php

// autoload.php @generated by Composer

require_once __DIR__ . '/composer/autoload_real.php';

return ComposerAutoloaderInit730293dc3aedabe296e80529404a4456::getLoader();
```

Mappage psr4

Fichier `autoload_psr4.php` :

```php
<?php

// autoload_psr4.php @generated by Composer

$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);

return array(
    'Twig\\' => array($vendorDir . '/twig/twig/src'),
    'Symfony\\Polyfill\\Mbstring\\' => array($vendorDir .
 ↪   '/symfony/polyfill-mbstring'),
    'Psr\\Log\\' => array($vendorDir . '/psr/log/Psr/Log'),
    'Monolog\\' => array($vendorDir . '/monolog/monolog/src/Monolog'),
    'MesClasses\\' => array($baseDir . '/src', $baseDir . '/lib'),
);
```

Mappage psr0

Fichier autoload_namespaces.php :

```php
<?php

// autoload_namespaces.php @generated by Composer

$vendorDir = dirname(dirname(__FILE__));
$baseDir = dirname($vendorDir);

return array(
    'Twig_' => array($vendorDir . '/twig/twig/lib'),
    'Symfony\\Component\\Validator' => array($vendorDir .
 ↪   '/symfony/validator'),
    'Doctrine\\DBAL' => array($vendorDir . '/doctrine/dbal/lib'),
    'Doctrine\\Common' => array($vendorDir . '/doctrine/common/lib'),
);
```