

# Programmation PHP

# Dans un gros projet ...

- on utilise souvent de nombreuses librairies pré-existantes,
  - exemple : [packagist.org](https://packagist.org), dépôt de packages PHP
- risque important de conflit de noms : plusieurs librairies utilisées dans 1 même projet risquent d'utiliser **le même nom** pour
  - des classes, des interfaces
  - des fonctions, des constantes

# réponse PHP : les *namespaces*

- Principe : les noms sont structurés dans une arborescence
- le nom complet d'une classe est préfixé par le namespace dans lequel elle est déclarée


```
<?php
namespace personapp\personne;

class Enseignant extends Personne { ...
}
```

The diagram illustrates the mapping of PHP class names to their fully qualified namespace paths. Arrows point from the class names **Enseignant** and **Personne** in the code to their respective paths: **\personapp\personne\Enseignant** and **\personapp\personne\Personne**.

```
<?php
namespace personapp\afficheur;

class AfficheurDEnseignant extends
    AfficheurDePersonne {
...
}
```



**\personapp\afficheur\AfficheurDEnseignant**

# utilisation

```
<?php
require_once 'Enseignant.php' ;
require_once 'AfficheurDEnseignant.php' ;

// Créer un enseignant :
$ens1 = new \personapp\personne\Enseignant( 'richards' ) ;

// Créer un afficheur
$aff1 = new \personapp\afficheur\AfficheurDEnseignant( $ens1 ) ;
```

# Définition des namespaces

- déclarés avec le mot-clé ***namespace*** obligatoirement placé avant tout autre code
- affecte **tous** les noms dans le fichier

```
<?php
namespace personapp\afficheur;

class AfficheurDEnseignant extends AfficheurDePersonne {
    public function __construct( Enseignant $p ) { ... }
    ERREUR : \personapp\afficheur\Enseignant

    public function __construct( \personapp\personne\Enseignant $p )
    { ... }
    OK : c'est le bon nom
}
```

- Dans un fichier contenant une directive namespace, tous les noms de classes/interfaces/fonctions/constantes,
  - appartiennent au même namespace
  - ou sont complètement qualifié (préfixé avec leur namespace de définition)
- note : les noms créés en dehors d'un namespace appartiennent au namespace \
  - par exemple : \Exception

```
<?php
namespace personapp\afficheur;

class AfficheurDEnseignant extends
    \personapp\afficheur\AfficheurDePersonne
{

    public function
        __construct(\personapp\personne\Personne $p ) {
        ...
    }

    public function __get( $attname ) {
        if (property_exists($this, $attname)) {
            return $this->$attname ;
        } else {
            throw new \Exception("invalid Property");
        }
    }

}
```



# alias de noms de classes

- les namespaces permettent d'organiser l'espace des noms et limitent les conflits de noms ...
- ... mais c'est pas très pratique d'utiliser systématiquement les noms complets de classes :
  - surcharge les programmes
  - risques d'erreurs dans les noms complets
  - fatigant pour les doigts de taper tout ça ;-)
- La directive **use** facilite l'utilisation de namespaces en permettant de définir des alias

```
<?php
namespace personapp\afficheur;

// alias de classe avec renommage
use \personapp\personne\Etudiant as Etu ;

// alias de classe sans renommage, raccourci pour
// use \personapp\personne\Personne as Personne ;
use \personapp\personne\Personne ;

// alias de namespace avec renommage
use \personapp\personne as p ;

class AfficheurDEnseignant extends AfficheurDePersonne {

    public function __construct( Personne $p ) { ... }

    public function afficherResultat( Etu $e ) { ... }

    public function collaboreAvec( p\Enseignant $e ) {
... }
```

# namespaces, répertoires et fichiers

- **bonne pratique** : organiser les répertoire et fichiers stockant 1 ensemble de classes de façon à ce que :
  - la partie *finale* de la hiérarchie des namespaces corresponde à la hiérarchie des répertoires,
  - la partie *initiale* (ou préfixe) de la hiérarchie des namespaces corresponde à un répertoire *racine*
- exemple : les classes du namespace
  - `\personapp\personne\` se trouvent dans
  - `<dir>/src/personne/`
  - `\personapp\afficheur\` se trouvent dans
  - `<dir>/src/afficheur/`

- **Intérêt** : à partir du nom complet de la classe, on connaît l'emplacement du fichier correspondant.
  - la classe `\personapp\personne\Etudiant` se trouve dans le fichier : `<dir>/src/personne/Etudiant.php`
  - la classe `\personapp\afficheur\AfficheurDEtudiant` se trouve dans le fichier : `<dir>/src/afficheur/AfficheurDEtudiant.php`

chemin désignant 1 fichier  
avec des /

```
<?php  
require_once 'src/personne/Enseignant.php' ;  
require_once 'src/afficheur/AfficheurDEnseignant.php' ;  
  
// Créer un enseignant :  
$ens1 = new \personapp\personne\Enseignant( 'richards' ) ;  
  
// Créer un afficheur  
$aff1 = new \personapp\afficheur\AfficheurDEnseignant( $ens1 ) ;
```

namespace avec des \

# PSR-4 : lien namespace-répertoire

## PHP Standard Recommendation

- structuration des namespaces et des fichiers pour **faciliter le chargement de classes**
- principe :
  - un namespace contient un **préfixe** qui correspond à 1 répertoire **racine**,
  - le reste du namespace correspond à une hiérarchie de répertoires ancrée à la racine
  - une classe est définie dans 1 fichier nommé exactement comme la classe avec le suffixe ".php" (hormis la racine)

# examples

## namespaces

\A\B\C\Classe1  
\A\B\D\Classe2  
\A\B\E\Classe3  
\A\B\F\Classe3

\A\Z\X\Classe11  
\A\Z\W\Classe12  
\A\Z\E\Classe13

préfixe

racine

\A\B

→

src/mod1

\A\Z

→

src/mod2

src/mod1/C/Classe1.php  
src/mod1/D/Classe2.php  
src/mod1/E/Classe3.php  
src/mod1/F/Classe3.php

src/mod2/X/Classe11.php  
src/mod2/W/Classe12.php  
src/mod2/E/Classe13.php

## fichiers

# PSR-4 : détails

- Les classes et les espaces de noms entièrement qualifiés doivent disposer de la structure suivante :
  - `\<racine>\(<Espace de noms>\)*<Nom de la Classe>`.
- Chaque espace de noms doit avoir un espace de noms racine (racine = préfixe).
- Chaque espace de noms peut avoir autant de sous-espaces de noms qu'il le souhaite.
- Chaque séparateur d'un espace de noms est converti en `DIRECTORY_SEPARATOR` lors du chargement à partir du système de fichiers ("`\`" ou "`/`" suivant l'OS).
- Les classes et espaces de noms complètement qualifiés sont suffixés avec `".php"` lors du chargement à partir du système de fichiers.
- L'espace de noms racine (préfixe) peut différer du répertoire racine (répertoire de base)



# Exemples variés

Nom de classe complètement qualifié	Préfixe de l'espace de noms	Répertoire de base	Chemin du fichier résultant
\Acme\Log\Writer \File_Writer	Acme\Log\Writer	./acme-log-writer/lib/	./acme-log-writer/lib/File_Writer.php
\Aura\Web \Response>Status	Aura\Web	/path/to/aura-web/src/	/path/to/aura-web/src/Response/Status.php
\Symfony\Core \Request	Symfony\Core	./vendor/Symfony/Core/	./vendor/Symfony/Core/Request.php
\Zend\Acl	Zend	/usr/includes/Zend/	/usr/includes/Zend/Acl.php

# Exemples variés

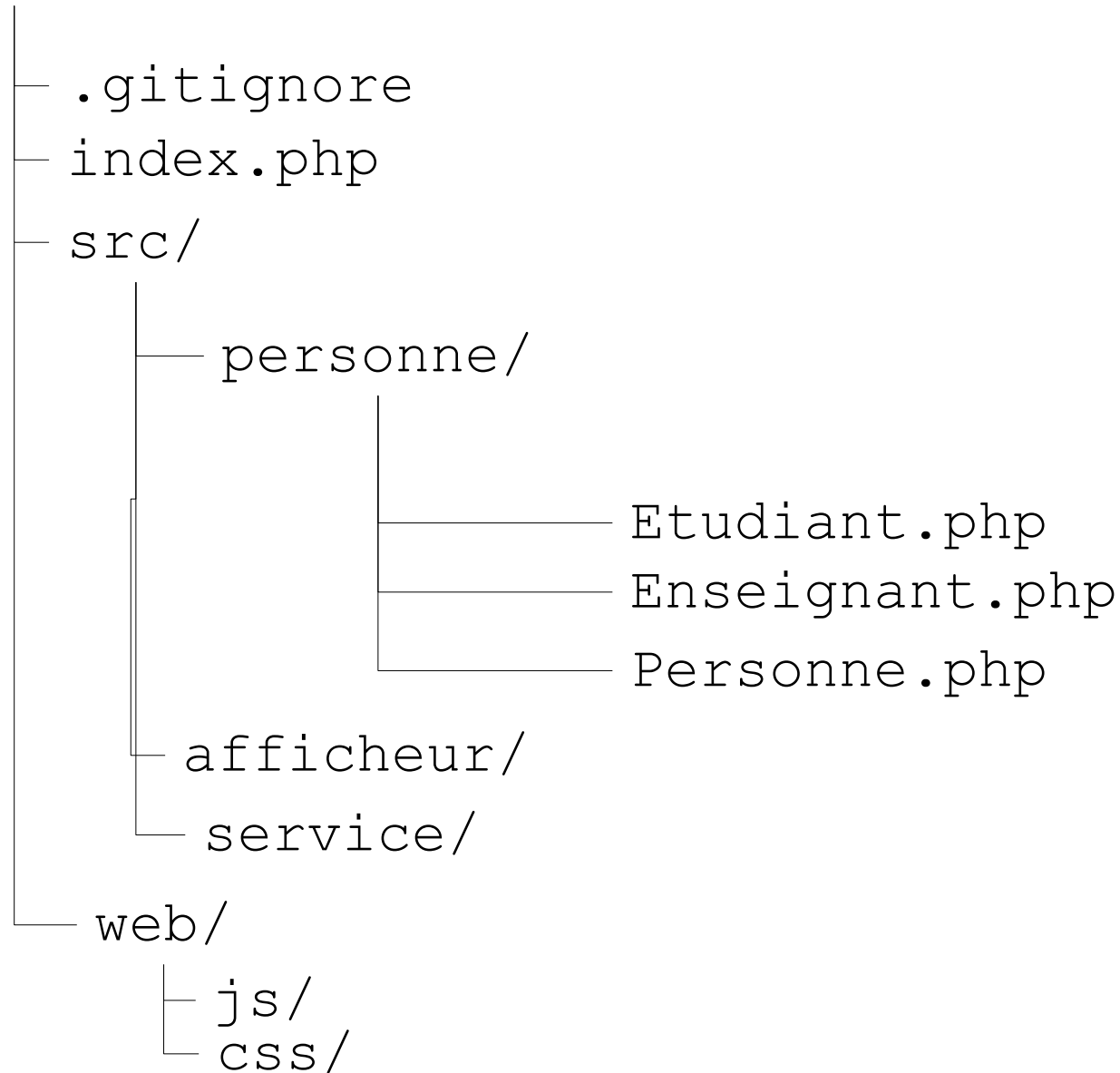
Nom de classe complètement qualifié	Préfixe de l'espace de noms	Répertoire de base	Chemin du fichier résultant
\myapp\models \User	myapp	./src/myapp/	./src/myapp/models/User.php
\Slim\Slim	Slim	./vendor/slim/slim/Slim/	./vendor/slim/slim/Slim/Slim.php
\Illuminate \Database \Eloquent\Model	Illuminate\Database	./vendor/illuminate/database/	./vendor/illuminate/database/Eloquent/Model.php

# PSR-4 : Préfixe et répertoire de base

- Nom de classe complet :
  - \Prefixe\espace\de\noms\Sous\espace\de\noms\MaClasse
- Fichier correspondant :
  - repertoire/du/prefixe/Sous/espace/de/noms/MaClasse.php
- - \Prefixe\espace\de\noms :  
correspond au répertoire de base repertoire/du/prefixe/  
(préfixe et répertoire de base peuvent être différents)
- - \Sous\espace\de\noms : correspond au chemin du répertoire de base  
vers le répertoire de la classe : Sous/espace/de/noms/  
(noms d'espaces et répertoires identiques)
- - Séparateurs d'espace de noms ("\") = séparateurs de répertoires  
(DIRECTORY\_SEPARATOR: "/" ou "\")
- - Nom du fichier = Nom de la classe + ".php"

# structuration des répertoires d'un projet

`<project_dir>`



```
<?php
```

```
require 'src/personne/Personne.php' ;
```

```
require 'src/personne/Etudiant.php' ;
```

```
require 'src/personne/Enseignant.php' ;
```

```
require 'src/afficheur/AfficheurDEtudiant.php';
```

**index.php**

- les scripts font les inclusions de tous les fichiers de classes nécessaires