

Programmation PHP

Dans un gros projet ...

- on utilise souvent de nombreuses librairies pré-existantes,
 - exemple : 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 class names to their fully qualified namespace paths. An arrow points from the class name **Enseignant** to the path **\personapp\personne\Enseignant**. Another arrow points from the class name **Personne** to the path **\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`

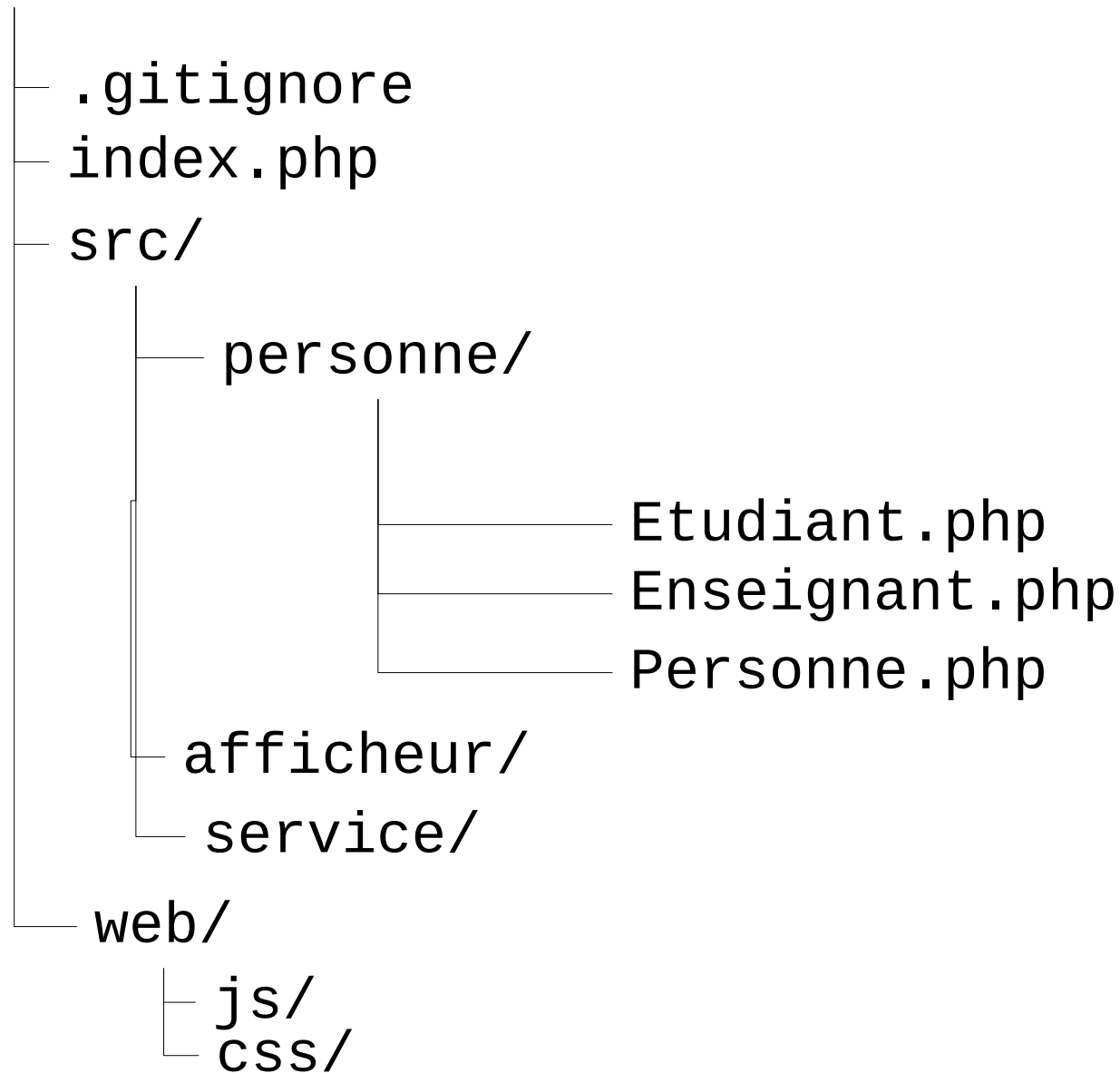
chemin désignant 1 fichier
avec des /

```
<?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 \

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