PHP : Standardisation

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

PHP Framework Interoperability Group

PHP-FIG

Groupe de travail visant à

- Normaliser le développement PHP.
- Harmoniser les pratiques.
- Faciliter la prise en main de cadriciels.

Fonctionnement inspiré du W3C et de l'IETF.
Emet des normes (PHP Standard Recommendation - PSR) :

- Auto-chargement et espaces de noms.
- Interfaces d'enregistrement d'historique (logging).
- Interfaces de mise en cache de données.
- Interfaces de représentation des messages HTTP.
- Style de codage.

Conventions de codage

Faciliter la prise en main de codes sources développés par d'autres programmeurs

- Organisation et contenu des fichiers.
- Nommage des variables, fonctions, classes, ...
- Indentation et dispositions des blocs (`if-else`, ...).
- Documentation, ...

Recommandations du PHP-FGI

PSR-1 et PSR-2 (Accepted), PSR-12 (Review).

Recommandations PSR-1 et PSR-2

Fichiers

Encodage UTF-8 sans BOM (indicateur d'ordre des octets).
Un fichier

- Déclare des symboles : fonctions, classes, ...
- OU ALORS est à effet de bord : inclusion de fichiers, affichage, modification des réglages de `php.ini`, accès à ressources externes, ...

Une classe par fichier figurant dans un espace de nom

Casse à utiliser :

- `StudlyCaps` pour les classes.
- `MA_CONSTANTE` pour les constantes de classes.
- `camelCase` pour les méthodes de classes.
- Format libre pour les propriétés d'objets et les fonctions (eg. `ma_fonction`).

Autres recommandations

Dénomination des variables et méthodes

- Les noms sont courts et explicites.
- Les noms de variables et fonctions ne contiennent que des lettres minuscules et le symbole souligné `'_'`.
- Un nom de variable est mis au pluriel s'il contient plusieurs éléments.
- Une méthode qui retourne un attribut d'objet commence par `get_`.
- Une méthode qui fixe un attribut d'objet commence par `set_`.

Autres recommandations

Disposition des blocs (coding_convention1.php)

```php
function divise(int $a, int $b) : float {
    if ($b==0) {
        throw new Exception("zero");
    } else {
        return $a/$b;
    }
}
```

Autres recommandations

- Supprimer le plus possible les espaces inutiles.
- Déclarer les variables au moment où on les utilise.
- Sortir des boucles les calculs inutiles ou redondants (refactoring).
- Typer et/ou documenter les fonctions, classes, méthodes, propriétés et variables.

Documentation

A quoi sert la documentation du code ?

Expliquer au programmeur qui ne connaît pas le code ce que :

- Contiennent les variables.
- Font les classes/méthodes/fonctions : rôle, comportement, paramètres en entrée/sortie, valeur de retour.

D'autant plus important en PHP étant donné l' absence de typage (eg. propriétés d'objets) et la non distinction entre procédure et fonction.

Documentation et commentaires

Commentaires

Trois types de commentaires :

```
// sur une ligne
/* sur plusieurs
lignes */
/**
* pour documentation
* (DockBlock)
*/
```

Ne pas insérer de lignes entre le commentaire et la variable ou la fonction à laquelle il se rapporte.

Génération de documentation avec PhpDoc

PhpDoc

Similaire à doxygen, Javadoc ...

- Permet de générer une documentation HTML à partir de fichiers sources PHP.
- Par insertion de `DocBlocks` dans le code source.

Générer la documentation (`phpdoc_example`)

```
$ phpdoc -s -d sourcedir -t targetdir
```

Format des commentaires

Balises génériques

On peut introduire des mots-clés (*tags*) au sein des commentaires qui seront interprétés pour structurer la documentation HTML :

- `@author` : nom des auteurs.
- `@copyright` : information de copyright.
- `@deprecated` : élément obsolète.
- `@license` : URL / nom.
- `@link` : URL / description.
- `@version` : description.

Format des commentaires

Autres tags

- `@param` : type d'argument.
- `@return` : type de valeur retour.
- `@see` : référence.
- `@todo` : description.
- `@var` : type de variable de classe.

Commentaire des classes

comment-class.php

```php
/**
 * classe utilisée pour représenter une personne
 * une personne est définie par son nom sous forme de
 * chaîne de caractères
 * @source 1
 */
class Personne {
   /**
     * nom de la persone
     *   @var string
     */
   protected $nom;
}
```

Commentaire des fonctions

comment-function.php

```php
/**
 * calcul de la moyenne
 * on calcule la moyenne d'un ensemble de notes donné
 *
 * @param array $arr_notes tableau de notes entières
 * @return float moyenne des notes
 */
function moyenne($arr_notes) {
    $nbr=count($arr_notes);
    $somme=0;
    // parcours séquentiel du tableau suivant la clé entière
    for ($i=0;$i<$nbr;++$i) {
        $somme+=$arr_notes[$i];
    }
    return $somme/$nbr;
}
```