

**Rédaction : OUKHENNICHE Abdelkrim**

**Date : 16/04/2017**

**Article N°201**

**Charte de codage**

**Alger le 16/04/2017**

**Article N°201**

**Charte de codage**

**Rédaction : OUKHENNICHE Abdelkrim**

**Article N°201**

**Charte de codage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Rédacteur | Description | Date | Validation |
| 1.0 | OUKHENNICHE Abdelkrim | Création du document | 12/04/2017 | TAKLIT Zina  IHADADENE Thinhinane |
| 1.1 | OUKHENNICHE  Abdelkrim | Modification du document | 13/04/2017 | IHADADENE Thinhinane |
| 1.2 | IHADADENE Thinhinane  TAKLIT Zina | Mise en page | 22/05/2017 | IHADADENE Thinhinane  YEBDA Sadia  TAKLIT Zina |

Tables de versions

Tables de matières

[Introduction 3](#_Toc483229213)

[1. Conventions de nommage 4](#_Toc483229214)

[1.1. Le nommage des fichiers code et des packages 4](#_Toc483229215)

[1.1.1. Les fichiers : XML, HTML, CSS et JS(Client) 4](#_Toc483229216)

[1.2. Les autres fichiers: PHP, Python,NodeJs(JsServer),Ruby... 4](#_Toc483229217)

[1.2.1. Règles générales 4](#_Toc483229218)

[1.2.2 Clas 5](#_Toc483229219)

[1.3. Le nommage des Packages 5](#_Toc483229220)

[1.4. Conventions des noms 5](#_Toc483229221)

[1.4.1. Noms adaptés 5](#_Toc483229222)

[1.4.2. Les variables 5](#_Toc483229223)

[1.4.3. Les constantes 6](#_Toc483229224)

[1.4.4. Les classes (Nom des classes) 6](#_Toc483229225)

[1.4.5. Fonctions et méthodes 6](#_Toc483229226)

[2. Conventions sur la lisibilité du code 7](#_Toc483229227)

[2.1. Structure du code 7](#_Toc483229228)

[2.1.1. La longueur de lignes 7](#_Toc483229229)

[2.1.2. Limiter la taille des modules 7](#_Toc483229230)

[2.1.3. L’indentation 7](#_Toc483229231)

[2.1.4. Les structures de contrôles 7](#_Toc483229232)

[2.2. Les commentaires dans le code 7](#_Toc483229233)

[2.2.1. Les commentaires 7](#_Toc483229234)

[2.2.2. Autres commentaires 7](#_Toc483229235)

[3. Conventions sur les bases de données 8](#_Toc483229236)

[3.1. Règles de nommage 8](#_Toc483229237)

[3.1.1. Noms de base de données 8](#_Toc483229238)

[3.1.2. Noms de tables 8](#_Toc483229239)

[3.1.3. Noms de colonnes 9](#_Toc483229240)

[3.1.4. Autres conventions 10](#_Toc483229241)

[Conclusion 10](#_Toc483229242)

# Introduction

Les règles de codage sont un ensemble de règles à suivre pour uniformiser les pratiques de développement logiciel, diffuser les bonnes pratiques de développement et éviter les erreurs de développement "classiques" au sein d'un groupe de développeurs.

Les règles de codage s'articulent autour de plusieurs thèmes, les plus courants étant :

* Le nommage et l'organisation des fichiers du code source ;
* le style d'indentation ;
* Les conventions de nommage, ou règles de nommage ;
* Les commentaires et documentation du code source ;
* La déclaration des variables ;
* Recommandations sur l'écriture des structures de contrôle.

# Conventions de nommage

## Le nommage des fichiers code et des packages

### Les fichiers : XML, HTML, CSS et JS(Client)

#### Version originale

Pour le nom d’un fichier code original :

* Utiliser uniquement [a-z][A-Z] [0-9][\.\_-] ;
* Pas d’espace ;
* Si le nom de fichier contient plus d’un mot ils seront séparés par des points.

#### Version optimisée

Si le nom de fichier original est nom.fichier.ext , le nom de fichier optimisé est nom.fichier.min.ext.

Avec ext = XML |HTML|CSS|JS

*Le fichier optimisé : est l’image de fichier original mais avec une taille plus petite en éliminant (les sauts de lignes, les commentaires...),concaténation, pour diminuer le trafic sur le site et aussi pour le chargement rapide des pages web.*

**Exemple**

|  |  |  |
| --- | --- | --- |
| Nom de fichier original | Nom de fichier optimisé | Extension |
| style.index.css | **style.index.min.css** | CSS |

## Les autres fichiers: PHP, Python,NodeJs(JsServer),Ruby...

### Règles générales

Pour les autres fichiers :

* Choisir des noms significatifs, qui indiquent ce qu’est la chose nommée ;
* Il vaut mieux des noms longs et clairs que courts mais abscons;
* Eviter les noms trop courts et trop longs ;
* Séparer les mots dans un nom en utilisant la norme lowerCamelCase.

**Exemple**

|  |  |
| --- | --- |
| Nom fichier | Type |
| connexion.php | Script PHP |
| loadView.py | Script Python |
| bibGestionClient.c | Code C |

### Clas

Pour le nom d’un fichier code de classe il faut :

* Les noms des fichiers doivent être en lowerCamelCase ;
* Utiliser uniquement [a-z] [A-Z] [0-9] ;
* Pour une classe PHP on utilise la norme nomDeLaClasse.class.php .

**Exemple**

|  |  |  |
| --- | --- | --- |
| Nom de la classe | Nom de fichier | Type |
| Client | **client.class.php** | PHP |
| AbonnementClient | **abonnementClient.class.php** | PHP |
| AbonnementClient | **abonnementClient.java** | Java |

## Le nommage des Packages

Pour le nom d’un package :

* Tout en minuscule ;
* Utiliser seulement [a-z] [0-9] ;
* Utiliser la norme lowerCamelCase ;
* Ne pas utiliser de tiret,de point,'-', '\_', ou d'autres caractères ($, \*, accents, ...).

## Conventions des noms

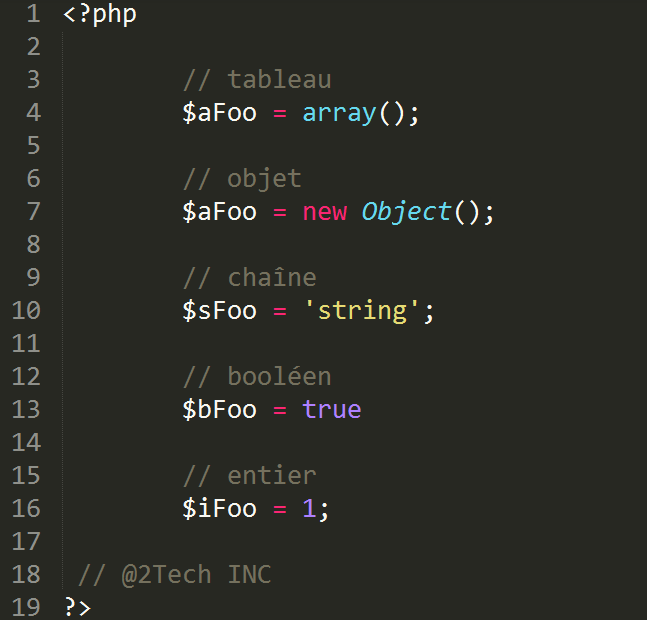
### Noms adaptés

Le nommage des différents éléments est crucial pour la compréhension du code. Un nom est le résultat d'un long processus de pensée résultant du sujet étudié. Si les noms sont appropriés alors tous les éléments s'adaptent ensembles naturellement, les programmeurs peuvent donc comprendre les rapports et leurs significations.

### Les variables

Pour une meilleure lisibilité il faut :

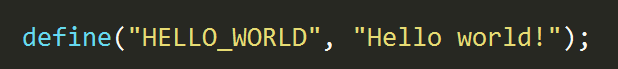
* Donner des noms simples et descriptifs ;
* Utiliser la norme lowerCamelCase ;
* Doivent être systématiquement déclarées avant leur utilisation ;
* Obligatoirement préfixées par une lettre (en majuscule) définissant leur type ;
* Utiliser seulement [a-z][A-Z][0,9][\_].

**Exemple**

### Les constantes

Les constantes doivent être :

* Ecrites en majuscule ;
* Chaque mot doit être séparé d'un simple souligné (\_) ;
* Utiliser seulement [A-Z][0,9][\_].

****

**Exemple**

### Les classes (Nom des classes)

Les noms des classes doivent être :

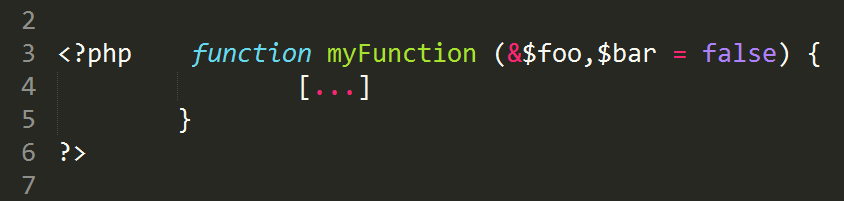
* En UpperCamelCase ;
* Eviter les abréviations lorsque cela est possible ;
* Les classes doivent avoir un nom parlant ;
* Utiliser uniquement [a-z][A-Z][0-9] ;
* Le nom d'une classe n'est jamais un verbe.

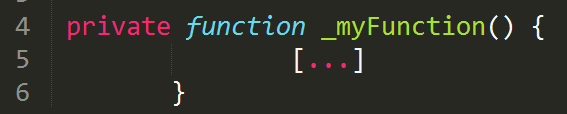
**Exemple :**

* Abonnement ;
* DossierClient ;
* ConnexionBaseDeDonnees.

### Fonctions et méthodes

Habituellement chaque méthode et fonction effectue une action, ainsi le nom doit être :

* Explicite afin de savoir à partir du nom ce qu'elle fait ;
* Utiliser des groupes verbaux ;
* Les fonctions et les méthodes doivent être nommées en lowerCamelCase ;
* Les arguments possédant des valeurs par défaut vont à la fin de la liste des arguments;
* Utiliser uniquement [a-z][A-Z][0-9] ;



* Pour les méthodes privées d'une classe, elles sont précédées d'un simple souligné (\_).

# Conventions sur la lisibilité du code

## Structure du code

### La longueur de lignes

Pour que le code puisse être imprimé et lu sur un écran standard, il faut limiter le nombre de caractères par ligne. Un choix courant est 80 ou 100 caractères maximum.

### Limiter la taille des modules

Pour les modules code:

* Pas plus de 50 ou 100 lignes de code pour une fonction ;
* Ecrire des fonctions courtes et simples ;
* Pas plus d’un millier de lignes dans un fichier code.

### L’indentation

L'indentation doit être de 4 espaces (ou 1 tabulation). Il n'y a aucune règle arbitraire quant au niveau d'indentation maximum, par contre si le niveau d'indentation est plus de 4 ou 5, il est préférable de factoriser le code si cela est possible.

### Les structures de contrôles

Les instructions de contrôle doivent avoir un espace entre le mot clé de l'instruction et la parenthèse ouvrante, afin de ne pas les confondre avec des appels de fonction. Il faut toujours utiliser des parenthèses, même dans les situations où elles sont techniquement optionnelles. Leur présence augmente la lisibilité du code et réduit le risque d'erreur logique lors de l'ajout de nouvelles lignes de code.

## Les commentaires dans le code

### Les commentaires

Pour les commentaires il faut suivre ces normes :

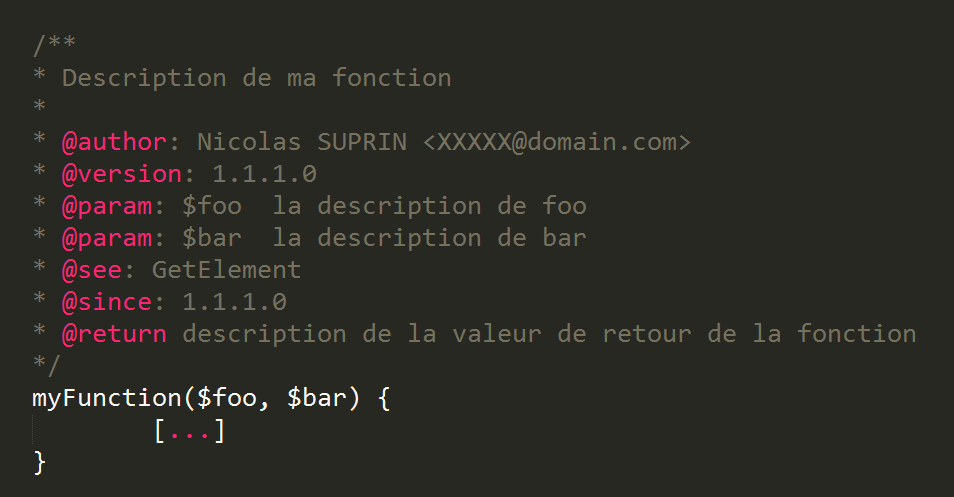
* Un code non commenté ou un code trop commenté, c’est le mal ;
* Commenter tout ce qui gagne à l’être, pas ce qui est clair ;
* Ajouter un commentaire avant chaque fonction précisant le contrat de la fonction ;
* En règle général, indiquer dans les commentaires ce que fait le code, pas comment il le fait (le code doit être suffisamment clair pour que le « comment » se lise tout seul).

### Autres commentaires

Voici les paramètres les plus utilisés et leurs fonctions :

* @author : auteur de la classe, de la méthode ou de la fonction ;
* @version : version de la classe, de la méthode ou de la fonction ;
* @param : permet de commenter le paramètre ;
* @see : permet de faire référence à d'autres classes, méthodes ou fonctions ;
* @since : permet de spécifier la compatibilité ;
* @return : même fonctionnement que param, mais pour décrire ce que renvoie la fonction ou la méthode.

**Exemple**



# Conventions sur les bases de données

## Règles de nommage

### Noms de base de données

Le nom d’une base de données doit être :

* Préfixer toutes les bases de données avec le préfixe BDD;
* Utiliser uniquement [a-z][0-9]ou simple souligné (\_).

**Exemple**:BDD\_messagerie.

### Noms de tables

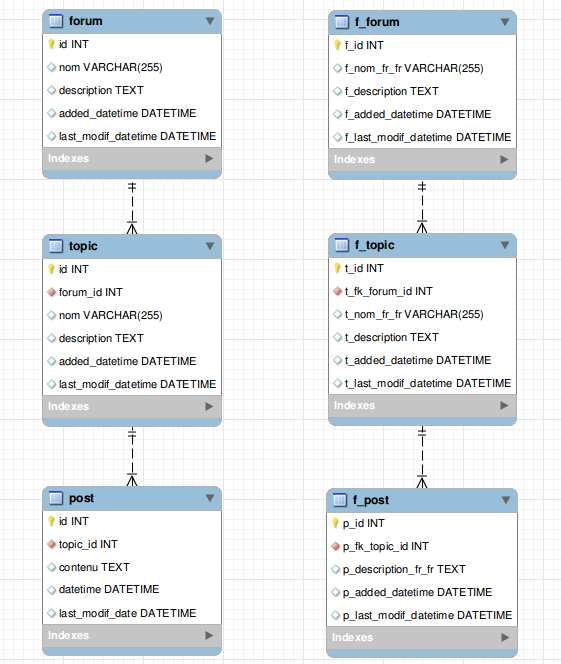
Pour les noms des tables il faut suivre ces normes :

* Utiliser un nom représentatif du contenu ;
* Utiliser un seul mot lorsque c’est possible ;
* Penser à des noms génériques et envisager les futures évolutions ;
* **Exemple** :Une table « client » qui pourrait aussi contenir les prospects et commerciaux devrait plutôt s’intitulé « utilisateur » ;
* Utiliser uniquement [a-z][0-9][\_].

### Noms de colonnes

Le nom des colonnes doit être :

* Préfixer toutes les colonnes de la même façon pour chaque table ;
* Lorsqu’une clé étrangère est utilisée il faut l’indiquer dans le nom de la colonne. La colonne peut contenir le préfixe, puis « fk » pour Foreign Key, puis le nom de la table et enfin se terminer par « id ». Ainsi, une colonne pourrait s’intituler « wp\_fk\_user\_id » (cf. préfixe « wp », foreign key sur la table utilisateur de la colonne « id ») ;
* Toujours intitulé de façon similaire certains champs tels que les DATE ou DATETIME.
* Utiliser uniquement [a-z][\_].

**Exemple** :BDD\_forum

### Autres conventions

Il faut suivre ces conventions suivantes :

* Ne pas utiliser les mots réservés.

**Exemple**:il faut éviter de nommer une colonne « date » car ce mot est déjà utilisé ;

* Ne pas utiliser de caractères spéciaux ;
* Eviter les majuscules. Pour écrire 2 mots il faut privilégier l’utilisation d’un simple souligné (\_).

**Exemple**:il faut plutôt utiliser « date\_inscription » que «DateInscription» ;

* Eviter l’utilisation d’abréviation.

# Conclusion

La vérification d'un ensemble de règles de programmation est généralement faite lors d'une revue de code ou bien d'un audit de code, manuellement ou bien automatiquement afin de trouver les violations de règles de programmation, dans le but de les corriger.

La revue de code vérifie entre autres qu'un nouveau code source est conforme aux règles de programmation avant que ce nouveau code source n'intègre la base de code existante.

L'audit de code vérifie que la base de code existante est conforme aux règles de programmation.