ATD QUART MONDE
AGIR TOUS POUR LA DIGNITÉ : REFUSER LA MISÈRE AVEC CEUX QUI LA VIVENT

Atos

SOLUTION D'ARCHIVAGE ÉLECTRONIQUE

Audit et bonnes pratiques de codage

Client :	ATD Quart Monde
Projet :	Solution d'archivage ATD Quart Monde
Référence :	
Version :	0.0
Date :	16 05 2017
Etat :	Draft
Classification :	Atos - Confidentiel

Circuit de validation

	Nom	Société	Date	Visa
Rédigé par :	Driss ALLAB	Atos	16/05/2017	
Vérifié par :				
Approuvé par :				
Validé par :				

Diffusion

Liste de diffusion		
Société/entité	**Prénom**	**Nom**

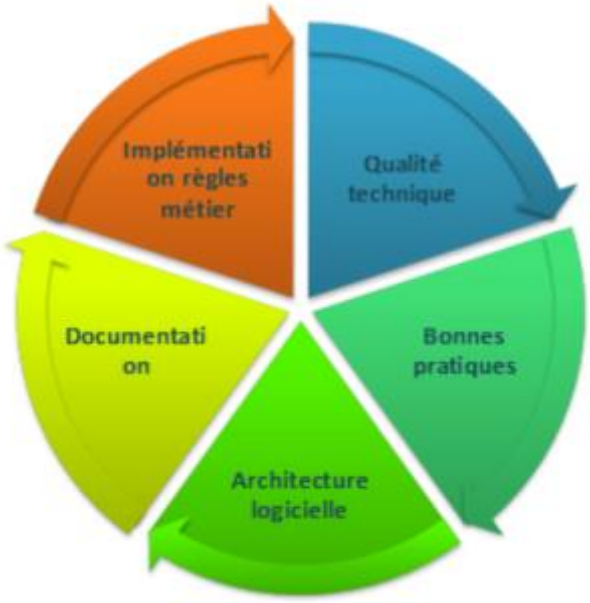
Historique des évolutions

Version	Date	Commentaires
0.0	16/05/2017	Version crée

Table de Matière

1 INTRODUCTION

Le développement du projet ATD quart monde est fait en PHP, Il est donc indispensable de respecter l'audit et les bonnes pratiques du PHP parce que les développeurs dans l'équipe peuvent être remplacés par des autres, et donc il faut respecter les standards et les normes d'utilisation de PHP et de la programmation web en générale pour permettre aux futurs développeurs de comprendre le code facilement.



Ce document est un aide-mémoire qui dresse une liste (non exhaustive) des points à contrôler dans le cadre d'une revue de code

2 Les pratiques du PHP

- **Mettre les chaînes de caractères entre simple quotes '...'** :
  Le traitement est plus rapide qu'avec les doubles quotes "..." car l'interpréteur PHP doit vérifier s'il y a des variables entre les doubles quotes.

- **Utiliser des quotes (simples, de préférence) pour accéder aux valeurs d'un tableau**
  Sinon PHP pense qu'il s'agit d'une constante.Exemple : $row['id'] est 7 fois plus rapide que $row[id]

- **Utiliser echo** qui est plus rapide que print

- **Faire un unset ou mettre à NULL les variables** qui ne sont plus utilisées, en particulier les gros tableaux.

- **Utiliser str_replace** pour remplacer les occurrences dans une chaîne. Il est plus rapide que preg_replace et globalement le meilleur dans tous les cas,

- **Définir une variable avant de l'utiliser** : incrémenter une variable NON définie est 9-10 fois plus lent qu'incrémenter une variable prédéfinie.

- **Privilégier les méthodes en static**. Elles sont 4 fois plus rapides.

- **Préférer la méthode POST à la méthode GET** quand c'est possible. La méthode GET ajoute les données à l'URL ce qui peut constituer un point de vulnérabilit

- **Ne jamais dévoiler d'informations concernant les chemins et configuration**

- **Ne jamais utiliser du texte clair pour stocker les mots de passe ou les comparer.**

- **Préférer le stockage des données des utilisateurs en SESSION plutôt que dans les COOKIES**
  - COOKIES : les données résident côté client. Ils peuvent être forgés (créés de toute pièce), modifiés ou supprimés par le client
  - SESSION : les données résident sur le serveur, le client n'y a pas accès

- **Changer l'ID de session** : utiliser la fonction session_regenerate_id() lorsqu'un utilisateur se connecte afin de limiter le vol de session

- **Fermer les connexions aux SGBD (Systèmes de Gestion de Bases de Données) après les avoir utilisées.**

3 Les pratiques HTML

- **Spécifier l'encodage** dans la balise meta des pages HTML (ex: <meta charset="utf-8" />)
- **Spécifier le type de document** dans chaque page HTML (ex: <!doctype html>)
- **Définir les emplacements pour le JavaScript**

  Si nous plaçons un script qui communique avec Google en début de page, entre les balises HEAD par exemple, avant d'afficher le contenu du site encadré par les balise BODY, il interagira avec Google.

  Par contre si on le place à la fin de la page, juste avant la balise de fermeture BODY, le navigateur affichera d'abord le contenu et à la fin de celui-ci, il enverra une requête à Google. le site s'affichera donc plus rapidement de cette façon pour le visiteur..

- **Mettre un attribut ALT à chaque balise d'image**

  L'attribut **ALT** de la balise **IMG** - image – n'as pas que pour but de faciliter la compréhension en cas de non affichage de cette image. il permet l'accessibilité par tous mais il améliore également le référencement.

```
<img src="img/photo.jpg" alt="Pierre en vacance" />
```

- **Préférer les feuilles de style CSS à l'attribut STYLE**

  Il est préférable et plus pratique de définir le style des éléments depuis la feuille de style CSS. Ainsi, **la page HTML sera plus clair et son indexation plus simple**.

4 Métriques

Quelques mesures destinées à fournir des indicateurs de la qualité du projet.

1. Longueur de classe

Les classes trop longues en font trop. Il s'agit de les restructurer afin qu'elles soient d'une longueur raisonnable. On procède alors à une opération de réusinage.
La limite communément admise est de 1000 lignes par classe grand maximum.

2. Longueur des méthodes

Quand les méthodes sont excessivement longues, l'excès d'information provoque une perte d'atten- tion du lecteur.
Restructurer votre code en créant, par exemple, d'autres méthodes ou d'autres classes d'une lon- gueur raisonnable.
Limite communément admise : maximum 100 lignes par méthode.

3. Nombre de méthodes

Une classe avec de trop nombreuses méthodes est une bonne cible de réusinage. Cela permet de réduire sa complexité et d'avoir des objets d'une granularité plus fine.
Limite communément admise : 10 méthodes par classe.

5 Normes de codage

- **Classes et Interfaces** : Anglais PascalCase (UpperCamelCase)
  Les noms de classe doivent correspondre au nom du répertoire sur le système de fichier qui contient le fichier de classe.
  Les noms doivent être alphanumériques uniquement, à l'exception d'un '_', qui doit être utilisé pour montrer la séparation dans le chemin.
  Si vous avez une classe fichier nommé Car.php dans le chemin '/ Modèles / Car.php' alors le nom de la classe devra être Models_Car.

- **Indentation du code**
  Il est préférable, pour la quasi totalité des langages objets, d'utiliser une indentation de 4 espaces, sans tabulation.
  Ceci permet d'éviter les problèmes avec les fichiers diff, les patches, l'historique CVS et les annotations.

- **Fonctions, Méthodes et Variables** : Anglais camelCase (lowerCamelCase)
  Des alphanumériques uniquement, à l'exception de '_' comme décrit ici.
  Le nom doit décrire le comportement de la fonction ou de la méthode / les données dans la variable.
  Les méthodes / variables qui sont déclarées avec modificateur privé ou protégé de visibilité doivent commencer par un

- **Constantes : ALL_CAPS**
  Des alphanumériques et '_' sont autorisés cependant '_' doit être utilisé pour séparer les mots dans des constantes.

- **Utiliser un fichier PHP par classe**.

- **Utiliser un fichier de configuration et de connexion à la base de données**

```php
<?php
$connect = mysql_connect('host','account','password');
//actions sur la BDD ?>
```

```php
include 'connect.php';
```

- **Documenter le code :** utiliser les commentaires en français pour expliquer le code et le fonctionnement du programme.

- **Toujours utiliser des accolades**, même dans les situations où elles sont techniquement optionnelles, pour augmenter la lisibilité

```php
echo "salut"; //affiche "salut"
#fin du programme
```

- **Lorsque vous ferez une requête, vous la ferez de la manière suivante :**

```php
mysql_connect(HOST, USER, PASS);
mysql_select_db(DB);
$temp = mysql_query('SELECT * FROM'. PREFIX.'ma_table');
```

6 Documentation du code

Il s'agit de conseils relatifs à la documentation du code en règle générale, indépendamment des langages de programmation.

1. Langue

Il est conseillé d'écrire la documentation en anglais (ainsi que les noms des classes, des méthodes, etc.) et ce, pour que le code puisse être repris par quiconque.

2. Classes

La déclaration d'une classe doit être précédée d'un commentaire destiné à expliquer son fonctionnement.

3. Méthodes

La déclaration d'une méthode, doit faire l'objet d'un commentaire explicite qui expose :

— Son fonctionnement, c'est à dire ce qu'elle fait ;

— Le type et le sens de chaque paramètre attendu ;

— Si la méthode est une fonction (et non une procédure), le commentaire doit exprimer le type

d'objet retourné et sa signification.

7 Conclusion

Adopter une façon de coder « standard » permet à tout le monde de se comprendre en lisant le même code. Il n'est pas nécessaire de la respecter à la lettre mais du moins ce qui est énoncé ici paraît le strict minimum à savoir.
De plus, coder d'une façon ou d'un autre n'a aucun effet sur le comportement du code, mais le faire proprement n'aura que des avantages.