# Clean Code

## L'ART DE BIEN CODER

# CHAPITRE 1

# A PROPOS

Cet ouvrage est offert gratuitement par le site Internet Dev Iot Yourself.

Vous pouvez librement le copier, le partager ou encore l'offrir en cadeau via un site Internet par exemple. Vous ne pouvez pas modifier le contenu sans autorisation préalable ni le vendre directement ou le partager dans des conditions non autorisées par la loi. **L'art de bien coder** est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas de Modification 3.0 non transposé. Les autorisations au-delà du champ de cette licence peuvent être obtenues en me contacter par le formulaire de contact de mon site Internet : http://www.deviotyourself.com/contact.

Si vous ne respectez pas les conditions décrites ci-dessus, le site Internet Dev Iot Yourself et son créateur se réserve le droit légitime de vous réclamer des dommages et intérêts.

# SOMMAIRE

2

# INTRODUCTION

Je suis Gaëtan Cottrez et ça fait maintenant plus de 10 ans que je code dont 6 ans dans le monde professionnel. J'ai toujours été passionné d'informatique dès le plus jeune âge mais c'est le développement, et plus précisément le web, qui m'a toujours attiré.

Analyser un besoin réel pour le développer  soi même  dans le but de faire gagner du temps et d'automatiser est quelque chose de très excitant et passionnant.

Comme beaucoup de développeur, je compte beaucoup sur ce que la communauté peut offrir pour m'aider à arriver à mes fins dans mes développements. Je me suis dit qu'il était temps pour moi aussi de rendre la pareil.

J'ai écris ce livre car je remarque qu'il y a beaucoup de personnes qui développe mais que peu d'entre eux peuvent se prétendre être des bons développeur. Tout le monde peut coder mais coder est un art et avoir les bases ne suffit pas.

Ce livre vous permettra d'obtenir les bons réflexes et les bonnes pratiques pour produire un code de qualité et cohérent avec votre projet de développement. Basé sur mon expérience et mon vécu, il vous apportera une approche réel de ce que doit être un développeur.

Je vous laisse le soin de le lire, de le découvrir, de le commenter et d'en discuter avec moi si vous le voulez par le biais de mon blog.

Je vous souhaite une très bonne lecture,

***Gaëtan Cottrez***

Dev Iot Yourself

CHAPITRE 3

# LES BONS REFLEXES POUR BIEN CODER

Les bons réflexes pour bien coder dès le départ pour ne pas prendre de mauvaises habitudes et s'enfoncer dans le « **dirty coding** ». Ce qui suit vous présente ce qu'un (futur) bon développeur doit absolument avoir comme réflexe et comme aptitude.

## N'arrêtez jamais d'apprendre

Le domaine de l'informatique de manière générale est **un domaine qui évolue très vite**, parfois trop. Cette constante évolution vous oblige à **mettre à jour vos connaissances et à continuer d'apprendre en permanence**, ce qui est une corvée pour certain et un plaisir pour d'autre. Sachez qu'un **développeur est obligé de faire de la veille technologique** surtout s'il ne veut pas se retrouver complètement dépassé par rapport au métier et surtout s'il ne veut pas voir sa carrière se terminer prématurément. Je ne vous le cacherais pas en vous disant que **si apprendre continuellement et intensivement de nouvelles connaissances, ce n'est pas pour vous alors vous n'excellerez jamais dans ce domaine**. Mais dîtes vous bien qu'**apprendre de nouvelles choses est passionnant** et que cela vous apporte un renouveau dans votre vie. **La routine n'existe pas chez le développeur.**

## Soyez autodidacte

La plupart du temps vous devrez vous cultiver en  cherchant à acquérir de nouvelles connaissances ou compétences par vous même. **La principale source d'enrichissement de vos connaissances sera Internet** bien évidemment. Ne mettez pas de côté les livres qui regorgent d'informations et de connaissances beaucoup plus précises et enrichissantes. SI vous cherchez une bonne édition de livre informatique, je ne peux que vous conseillez **des livres de l'édition ENI**  qui sont très bien fait avec des fichiers à télécharger complémentaires à ce qui se trouvent dans les livres. **Oubliez les livres «  Pour les nuls  »** qui vulgarisent trop souvent les domaines qu'ils abordent.

## Prenez des notes et réalisez des procédures de ce que vous faîtes

Si **vous ne prenez jamais de notes** sur ce que vous apprenez ou ce que vous faites, alors **il est temps de vous y  mettre dès maintenant**. Dîtes vous bien que **ce n'est pas du temps perdu** bien au contraire, ça vous permet de **mieux assimiler vos connaissances** et d'obtenir de la clarté dans ce que vous produisez. Il est toujours **plus facile de visualiser une connaissance ou une idée sur papier que dans son esprit**. Personnellement, j'ai un «white board» et je note tout ce que j'ai en tête. Comme il est dans mon bureau, j'ai toujours mes idées sous les yeux lorsque je développe.

## Partagez  ce que vous avez appris

C'est un **moyen efficace  pour vérifier vos connaissances** et par la même occasion de vous auto-évaluer. Croyez-moi **partager vos connaissances vous rendra un bien meilleur développeur** que vous êtes car **vous vous efforcez de vous appliquer dans votre partage, vous donnera une meilleure estime de vous même** et cela  **vous procurera une poussée de motivation** pour continuer votre progression.

## Restez simple et allez à l'essentiel

Rien ne sert de faire compliqué ! Je suis partisan de dire qu'**il faut commencer simple et étoffer par la suite**. Il est toujours plus simple et plus rapide de commencer un développement très simple afin de se définir une coquille applicative. Créer une coquille d'un développement est très important puisque cela vous permet de visualiser plus facilement ce que vous voulez obtenir.

## Participez à des conférences ou des événements

Il existe  **beaucoup de conférences gratuites** et souvent elles sont centralisées sur des  sites comme  meetup pour s'en rendre compte.  Il est clair que **les payantes sont les plus intéressantes** et les plus enrichissantes. Si vous aimez le PHP, je ne peux que vous conseillez le **PHP Tour**. Si vous êtes intéressés par tout ce qui touche au web, alors je vous conseille le **web2day**. Enfin une conférence que j'adore et qui possède un rapport qualité/prix imbattable c'est la **Dev Day**  à Mons (Belgique).

## Restez concentré

Les distractions, sous toutes leurs formes, seront votre ennemi. Que ce soit par les notifications de votre smartphone, vos notifications d'e-mails arrivant sur votre ordinateur, votre conjoint qui vous demande ce que vous voulez manger au dîner, vos collègues qui vous sollicitent un peu d'aide... Au final vous ne vous voyez jamais avancer. Je vous conseille de vous isoler le plus possible. Si vous ne pouvez pas, vous devez imposer des règles à vous-même mais aussi à votre entourage pour vous éloigner de vos distractions. Soyez discipliné et vous verrez par vous-même que le fait de rester très concentré vous rendra plus productif dans votre développement de manière générale.

## Surmontez vos problèmes intelligemment

Tôt ou tard, vous êtes confrontés à des problèmes plus ou moins compliqués et qui vous prennent pas mal de temps. L'obstination sera votre pire ennemie dans ce cas là. Si vous vous obstinez à passer du temps (plus d'une heure) à tenter de résoudre votre problème, vous ne serez pas productif. Quand vous bloquez sur un problème, il faut vous poser la question si celui-ci bloque le reste de votre développement et dans la majeur partie des cas, il ne le sera pas. Passez 20 min sur un problème et prendre la décision de le revoir plus tard est un bon moyen de le résoudre car inconsciemment votre esprit continue de travailler à ce problème tout en continuant votre développement. Vous aurez ainsi une illumination quelque temps après en vous disant : «Bon sang ! la solution à mon problème c'est ça.». Dans le cas où celui-ci est bloquant, n'hésitez pas si vous travaillez en équipe de la solliciter. «Plusieurs cerveaux en vaux mieux qu'un» et quelqu'un d'externe à votre problème pourra vous donnez des pistes fraîches pour le résoudre.

Petites astuces : lorsque vous cherchez une réponse ou solution à votre problème, consultez le site Internet stackoverflow où vous serez dans 99% que le problème que vous avez a déjà été abordé.

## Le plus important… Soyez passionné

« ***CHOOSE A JOB YOU LOVE AND YOU WILL NOT HAVE TO WORK A DAY IN YOUR LIFE*** », tel est ma devise. Dîtes vous bien que **la passion est l'élément le plus important pour devenir un bon développeur**. Cela vous procurera de l'**envie pour avancer, pour apprendre mais aussi de la motivation, de la satisfaction et de l'épanouissement personnel**.

CHAPITRE 4

# MÉTHODOLOGIE DE TRAVAIL POUR BIEN CODER

## Comprenez et analysez ce que vous devez faire avant de vous lancer

Il est très important quand vous avez une fonctionnalité à coder de **bien la comprendre et de l'analyser au préalable avant de commencer**. Sans un minimum de compréhension on risque de développer une fonctionnalité qui ne correspondra pas aux attentes, ce qui est frustrant et représentera une perte de temps.

N'hésitez pas à **noter sur papier votre fonctionnalité et à dessiner des schémas** si besoin pour pouvoir vous projeter.

## Coder en langage humain

On dit souvent que lorsque l'on demande une fonctionnalité à un **développeur** celui-ci c**ommence à traduire ce qu'on lui demande et à imaginer les lignes de code**. Si vous êtes dans ce cas là, c'est tout à fait normal mais **vous faîtes erreur en essayant de traduire ce qu'on vous demande**.

**Pourquoi un développeur devrait se fatiguer à tenter de comprendre la demande de son interlocuteur, la traduire dans son langage et pour, lorsqu'il aura terminer de la coder, la retraduire à son interlocuteur pour lui expliquer son fonctionnement ?** Cette chaîne de traduction rendra au final dans la majeur partie des cas une fonctionnalité erronée par rapport à la demande.

Vous devez **coder tout simplement dans un langage humain**. Il faut imaginez qu'une personne non expérimentée en programmation puisse comprendre et lire votre code si elle devait tomber dessus. Vous serez sûr que ce que vous codez respectera votre fonctionnalité. Retenez donc bien : **Dîtes vous en langage humain ce que vous devez faire et coder là de cette façon**.

## Découpez votre code

Il faut **éviter d'avoir des classes, méthodes ou même des fonctions avec trop d'instructions au niveau du code** car on peut vite s'y perdre à la lecture. Je vous recommande de **découper des grosses méthodes en plusieurs petites méthodes**. Le plus simple est de partir du principe qu'**une méthode** ne doit faire qu'**un seul traitement et rien d'autre**.

## Ne mélanger pas le français et l'anglais

Vous devez commencer à coder dans une seule langue et vous y tenir que ce soit vos noms de variables, vos méthodes ou classes. Il est important de **ne pas jongler avec 2 langues quand on code**. Personnellement je préfère **coder en anglais** car en plus d'apprendre du vocabulaire supplémentaire  j'aime avoir  un projet global et uniforme car tôt ou tard vous allez intégrer des fonctionnalités et librairies externes dans vos projets qui seront elles en anglais.

## Coder mal pour bien coder

Quand vous devez développer une nouvelle fonctionnalité et que vous n'êtes pas à l'aise avec les bonnes pratiques pour bien coder alors **coder mal MAIS** seulement si vous vous promettez de revenir sur ce code dans **un laps de**

**temps très court**. **Quand on est débutant ou face à l'inconnu**, comprendre la problématique, trouver le moyen de la mettre en place et réaliser la tâche en voulant bien **la coder provoque un sentiment de frustration et de panique**.

Prenons l'exemple d'un article de blog à rédiger : on **fixe le sujet** puis on **définit ses parties**, on **écrit comme on le pense** ses idées en morceau de phrase et avec des mots de clés pour chaque partie. **Puis on développe** ses idées en formulant des phrases claires puis on relit le tout et on peaufine les détails.

Le développement c'est pareil. On **fixe la fonctionnalité (le sujet)** puis on définit **son algorithme (ses parties)**, on code de manière non optimisée (on écrit comme on le pense) puis **on revoit son code pour l'améliorer** (on développe ses idées et on peaufine les détails)

Mon conseil : vous devez séparer l'art de bien coder et l'objectif de votre fonctionnalité. Vous devez commencer par vous concentrer sur cette dernière. Une fois celle-ci réalisée, vous aurez un algorithme pas optimale en face des yeux mais ce sera déjà plus clair dans votre esprit. Il ne vous reste plus qu'à revoir l'ensemble du code de la fonctionnalité et d'appliquer les bonnes pratiques pour un avoir un code clair. L'idéal est de le faire juste après avoir développé la fonctionnalité.

CHAPITRE 5

# STRUCTURER ET ORGANISER SON CODE

## Indentez votre code

Quelque chose d'évident mais fréquemment non respectée car **certaines parties de codes sont mal indentées et c'est toujours la partie sur laquelle on revient un jour ou l'autre**. La plupart des éditeurs de texte comme **Notepad ++**, **Sublime Text** ou bien **Atom** le font déjà mais ce que certains ne font pas (nativement) c'est la possibilité de ré-indenter du code mal indenté. Je vous conseille d'utiliser un **IDE** pour vos développements car en plus d'**améliorer votre productivité** ils contiennent généralement des fonctionnalités dont notamment la possibilité de ré-indenter correctement une portion de code.

## Privilégiez l'orienté objet

Je vais être clair avec vous. Peu importe les développements que vous ferez **vous aurez toujours besoin du procédural** mais **il faut penser en orienté objet** pas parce que c'est mieux juste parce **c'est une bonne façon de structurer et de penser son code**. Si vous ne connaissez pas l'orienté objet, je vous conseille de découvrir cette philosophie de coder qui améliorera la structure ainsi que le découpage de votre code. Il est d'ailleurs beaucoup plus facile de développer des fonctionnalités sous forme de classe.

## Fractionnez vos fonctionnalités en plusieurs parties

**Vous êtes entrain de coder une nouvelle fonctionnalité dans votre projet et celle-ci fait plus de 100 lignes ?** Alors **fractionnez là en plusieurs parties**, essayez de repérer des parties codes qui peuvent être délocalisées ailleurs que dans votre fonctionnalité principale.

## Adoptez une architecture  MVC

Si vous avez déjà utilisé un framework, vous avez sans doute remarqué que celui-ci est **construit de manière structurée**. Ils adoptent pour la plupart une structure MVC  séparant ainsi chaque processus d'un code : la «**Vue**» pour l'affichage des données à l'utilisateur, le «**Modèle**» pour les interactions avec votre  base de données et le «**Contrôleur**» pour gérer les fonctionnalités de votre application et faire le lien avec la vue et le modèle. C'est la façon la plus simple pour structurer son code. Chacun de ses processus étant physiquement séparés par des dossiers respectifs (**controller, model, view**)

Ainsi si vous avez besoin de c**omprendre un processus métier**, vous savez qu'il faut chercher dans **le contrôleur**. Une **erreur SQL** s'est produite ? C'est dans **le modèle** qu'il faut intervenir. Un **bug d'affichage** ? Allez voir dans **la vue**. C'est toujours plus facile quand on sait où chercher.

Il est également important d'**isoler dans un dossier séparé les ressources** dites  public. Ce que j'appelle une ressource ce sont tous vos  **fichiers images, CSS, JavaScript et bien d'autres**. Ce sont généralement des **fichiers interprétés par le client (navigateur)** et pouvant être facilement récupérable par l'internaute.

CHAPITRE 6

# AMÉLIORER LA LISIBILITÉ DE SON CODE

## Nommez correctement vos variables

Il faut toujours **nommer ses variables avec des noms précis**. Une simple lecture doit donner directement son intention et leur but. **Évitez donc les noms de variables peu précis** du genre :

```
1 // Adresse de facturation
2 $Adresse1 = "";
3 // Adresse de livraison
4 $Adresse2 = "";
```

Privilégiez plutôt :

```
1 $AdresseDeFacturation = "";
2 $AdresseDeLivraison = "";
```

C'est quand même plus clair comme ça non ?

Comme **les noms de variables sont explicites, les commentaires ne sont plus nécessaires**.

# Évitez de "hard coder"

"**Hard coder**" des valeurs **n'est jamais une bonne chose**. Cela **nuit à la lisibilité de votre code et** n'arrange en rien **sa maintenance**. Prenons l'exemple de ce code :

```
1 if($nb_abo > 0){
2     $val = $nb_abo * 35;
3 }else{
4     $val = 0;
5 }
```

Difficile d'évaluer le contexte facilement quand les noms de variables ne sont pas précis et on ne sait pas exactement **à quoi correspond la valeur 35**.

```
1 define('VALEUR_UNITAIRE_ABONNEMENT', 35);
2 define('VALEUR_DEPART_ABONNEMENT', 0);
3
4 $MontantAbonnementPris = VALEUR_DEPART_ABONNEMENT;
5 if(PriseDunAbonnement($nombre_abonnement_choisi)){
6    $MontantAbonnementPris = $nombre_abonnement_choisi * VALEUR_UNITAIRE_ABONNEMEN
7 }
```

Maintenant nous **identifions bien le contexte** et à quoi correspond **le 35** puisqu'elle a été isolée dans **une constante** qui a été **correctement nommée**. Grâce à cette constante, nous pourrons utiliser cette valeur ailleurs dans l'applica-

tion et si la valeur de l'abonnement doit changer il suffira de modifier la constante.

## Évitez les booléens en paramètre d'une méthode ou d'une fonction

Il est **déconseillé d'avoir comme paramètre des booléens** dans ses méthodes ou ses fonctions. Quand on lit la définition d'une méthode directement dans la classe, on peut comprendre l'intérêt du booléen :

```php
public function Calcul($multiplication,$valeur1, $valeur2){
    if($multiplication){
        // faire une multiplication
        return $valeur1 * $valeur2;
    }else{
        // faire une addition
        return $valeur1 + $valeur2;
    }
}
```

Mais lorsqu'on l'utilise, on ne comprend pas à quoi sert le paramètre :

```php
$class->Calcul(false,3,4);
```

Comme il est très intéressant de découper son code, il vaut mieux découper cette méthode en deux méthodes :

```php
public function Addition($valeur1, $valeur2){
    return $valeur1 + $valeur2;
}

public function Multiplication($valeur1, $valeur2){
    return $valeur1 * $valeur2;
}
```

Ce qui donne du code plus lisible  à l'utilisation :

```php
$class->Addition(3,4);
$class->Multiplication(3,4);
```

**Un conseil** : si vous ne savez pas faire autrement (car oui de temps en temps on n'a pas le choix), je vous conseille lors de l'appel d'une méthode ou d'une fonction qui contient en paramètre un booléen de **stocker la valeur «true» ou «false» dans un nom de variable précis** et d'utiliser cette variable en paramètre pour une meilleure lisibilité de votre code.

## Commentez quand cela est nécessaire

Si vous appliquez ces recommandations citées plus haut,  il sera fort probable que **vous ne soyez plus obliger de commenter votre code** comme auparavant.

Au lieu commenter pour expliquer vos variables, méthodes ou fonctions, **vous commenterez désormais que pour 2 choses** :

- exprimer vos intentions
- exprimer vos choix

Vos commentaires apporterons  réellement une plus-value à votre code. Voici un exemple :

```
public function chargement_des_parametres(){
    /*
     * J'ai pris la décision d'implémenter cette méthode
     * dans cette classe pour pouvoir y accéder depuis n'importe
     * autre classe qui sera étendu de celle-ci
     */

    (...)
}
```

**Encore une chose** : Si vous travaillez en équipe, **évitez de laisser du code que vous avez commenté**. Cela peut **mettre des  interrogations sur l'existence de ce code** commenté. Si bien entendu, vous ne pouvez pas faire autrement complétez ce code commenté par un commentaire de choix

CHAPITRE 7

# BIEN CODER EN TESTANT LA SÉCURITÉ DE SON

Tester la sécurité du code que l'on vient de produire. C'est une étape indispensable pour un développement... Normalement.

## Des failles en veux-tu, en voilà

Que ce soit un développement web ou un développement client lourd, **la plupart des logiciels que nous utilisons sont remplis de faille de sécurité**.  Ce qui est encore plus grave, c'est que la plupart des personnes les utilisant ne se rendent pas de la gravité de la chose. **Un bon développeur doit avoir bonne conscience pour ses futurs utilisateurs** et **produire un code de qualité en le sécurisant**.

## Ce n'est pas toujours la faute du développeur

On  dira que c'est souvent la faute du développeur parce c'est lui qui produit l'application. Inattention, manque d'expérience, aucun intérêt pour la tâche à effectuer... Les raisons sont vastes. Le rôle d'**un développeur c'est être attentif à la sécurité en se renseignant et en la mettant en place**. Relativisons tout de même car malgré ce qu'on pourrait croire **ce n'est pas toujours sa faute**. Dans un milieu professionnel, le timing ainsi que le budget pour un projet influe la façon de coder. Ces 2 facteurs influençant lourdement sur la qualité de votre code

produit et bien évidemment sa sécurité. Retenez bien que pour vos projets personnels vous avez le temps de produire du code propre et de le sécuriser.

## Des outils sont à votre disposition

Le meilleur moyen de tester la sécurité de son application c'est d'utiliser des outils de test de sécurité. J'utilise une multitude d'outils fournis dans une distribution Linux destiné à la sécurité : Kali Linux. Si vous faîtes du développement web, on y trouve pas mal d'outil comme :

- sqlmap (pour faire des tests d'injections SQL)
- wpscan (pour identifier et connaître les potentiels failles de sécurité sur un site Internet WordPress)
- PHP Vulnerability Hunter (qui comme son nom l'indique permet de détecter les failles de sécurités de votre application en PHP)
- Open Vas (qui n'est pas destiné au site web mais qui testera par tous les moyens de trouver une faille de sécurité, peu importe son type)

Il est possible d'obtenir ces outils autrement que par Kali Linux bien entendu.

## L'hébergement de votre application est important

Pour du développement web, **héberger son application web sur un serveur mutualisé est dangereux**. Même si le coût est intéressant, vous jouez la sécurité de votre application. En effet, **si le serveur est une passoire ou si une autre application non sécurisé est hébergé, vous risquez la sécurité de votre application**. Pour vous rendre compte de ce constat, il suffit de voir mon tu-

torial sur sqlmap pour vous rendre compte de la problématique du mutualisé. Privilégiez un serveur dédié et si vous le pouvez passez votre application web en HTTPS pour obtenir un chiffrement des données que vos utilisateurs généreront dessus.

# CHAPITRE 8

# CONCLUSION

Grâce à cet ouvrage, j'espère avoir répondu à certaines de vos interrogations et apporté plusieurs pistes sur la voie de bien coder. Il a été écrit sur base de mon expérience et de mes erreurs pour vous donner les éléments nécessaires pour adopter rapidement les bonnes pratiques pour bien coder.

Vous aurez beaucoup de difficultés à toutes les appliquer. La difficulté l'est encore plus quand vous codez dans le monde professionnel car en fonction du budget et du timing d'un projet, le dirty coding est souvent de main mise. Il est moi-même très difficile d'appliquer à la lettre tout ses bons principes dans le monde professionnel mais je me réconcilie avec moi-même dans mes développements personnels.

Certaines parties de cet ouvrage peuvent être plus complètes et plus précises. Vous pouvez me faire vos suggestions, me poser vos questions ou me demander des conseils en **me contactant** directement sur mon blog. Vous trouverez également de nombreux articles pouvant enrichir le contenu de cet ouvrage.

Je vous invite d'ailleurs à le visiter et à me suivre sur les réseaux sociaux **Twitter** ou **Facebook** !

Je vous dit à très bientôt et surtout coder bien !

*Gaëtan Cottrez*

[Dev Iot Yourself]