

# ETML : Conventions de codage

Version 3.6.0

## Table des matières

<b>1</b>	<b>BUT .....</b>	<b>3</b>
<b>2</b>	<b>CONSIDÉRATIONS GÉNÉRALES.....</b>	<b>3</b>
2.1	Documentation .....	3
2.2	Algorithmes.....	3
2.3	Commentaires .....	4
2.4	Lisibilité.....	5
<b>3</b>	<b>RÈGLES DE CODAGE .....</b>	<b>6</b>
3.1	Entête .....	6
3.2	Nommage.....	7
3.2.1	Général.....	7
3.2.2	Constante .....	7
3.2.3	Variable .....	7
3.2.4	Les tableaux et les listes.....	7
3.2.5	Méthode.....	7
3.2.6	Propriété .....	7
3.2.7	Classe.....	8
3.2.8	Interface (au sens POO) .....	8
3.2.9	Composants graphiques (UI) .....	8
3.2.10	Bases de données .....	9
<b>4</b>	<b>GESTION DE VERSION .....</b>	<b>10</b>
4.1	Numéros de version .....	10
<b>5</b>	<b>APPRENDRE PAR L'EXEMPLE.....</b>	<b>11</b>
5.1	C#.....	11
5.2	PHP .....	12
5.3	JavaScript.....	12
5.4	HTML.....	13
5.4.1	Structure des dossiers .....	13
5.4.2	Code .....	14
5.5	CSS .....	15

## 1 But

Ce document regroupe les règles de codage à l'ETML concernant tout travail lié à du code, que ce soit :

- Des projets internes ou externes
- Des exercices
- Des projets personnels réalisés dans le cadre de l'apprentissage
- Des scripts ou utilitaires développés pour la formation

Il n'est pas nécessaire de les appliquer pour du code auto-généré ou des bibliothèques déjà écrites par des tiers.

Néanmoins, les sources doivent être mentionnées :

```
/// <summary>
/// Convertit une température donnée en fahrenheit vers des degrés Celsius
/// inspiré de http://www.mathsisfun.com/temperature-conversion.html
/// </summary>
/// <param name="temperature">la température en fahrenheit</param>
/// <returns>la température convertie en degré Celsius</returns>
/// <exception cref="Exception">Si la température ne peut être convertie</exception>
public float ConvertToCelsius(float temperature)
{
    const float FARENHEIT_TO_CELSIUS_RATIO = 5 / 9;
    const int FARENHEIT_TO_CELSIUS_DELTA = -32;

    return (temperature + FARENHEIT_TO_CELSIUS_DELTA) * FARENHEIT_TO_CELSIUS_RATIO;
}
```

## 2 Considérations générales

### 2.1 Documentation

Selon la demande du formateur ou du client, une documentation particulière peut être demandée (schémas de base de données, diagrammes UML, maquettes d'interface) en plus du code source.

Dans tous les cas, il est important de garder à l'esprit que le code est ce qui va naturellement évoluer. Ainsi les commentaires liés à ce code doivent donc être pris en compte avec vigilance.

### 2.2 Algorithmes

Les algorithmes doivent être choisis selon les critères suivants

- Réduction des risques de bug
- Facilité d'évolution
- Optimisation de l'utilisation des ressources
- Facilité à comprendre

## 2.3 Commentaires

Commenter le code est un bon investissement pour réduire le coût des applications puisque cela facilite sa compréhension pour les personnes impliquées dans la maintenance ou l'évolution du programme.

C'est pourquoi, le but des commentaires est surtout d'expliquer les éléments sujets à une interprétation confuse.

A ce titre, il est recommandé de commenter les **groupes d'instructions** logiques plutôt que de commenter chaque ligne de code (exception faite pour des raisons pédagogiques).

Les commentaires peuvent être écrits en français ou en anglais.

*Exemples :*

### *Commentaires pertinents*

```
/// <summary>
/// Cette méthode est un bon exemple comprenant
/// des commentaires pertinents
/// Elle n'est pas censées être utilisée en production
/// </summary>
private void UsefullComments()
{
    //Validé par l'académie Jedi du 24 juin 4096
    //(voir règlement #JS2712)
    const int MIN_POWER_TO_BE_A_JEDI = 5;

    int availablePower = 10;

    //Par défaut, un humain n'est pas considéré comme un jedi
    //(voir règlement #12)
    bool isAJedi = false;

    //Ceci vérifie la loi Jedi.
    //A modifier avec précaution pour éviter un déséquilibre
    //équivalent à ce que la planète terre a connu en 2028
    if (isAJedi && availablePower > MIN_POWER_TO_BE_A_JEDI)
    {
        Console.WriteLine("Congrats, you're a jedi");
    }
}
```

### *Commentaires descriptifs (utilité réduite sauf pour favoriser l'apprentissage)*

```
/// <summary>
/// Cette méthode est inutile
/// </summary>
private void UselessComments()
{
    //La puissance minimale pour un jedi
    const int MIN_POWER_TO_BE_A_JEDI=5;

    //La puissance disponible
    int availablePower = 10;

    //Il n'est pas un jedi
    bool isAJedi = false;

    //Si je suis un jedi et que ma puissance est plus
    //grande que 5 alors...
    if (isAJedi && availablePower > MIN_POWER_TO_BE_A_JEDI)
    {
        Console.WriteLine("Congrats, you're a jedi");
    }
}
```

## 2.4 Lisibilité

- Le code doit être lisible, aéré et indenté
- Les constantes doivent être utilisées partout où c'est possible
- Le code doit être découpé en fonctions/méthodes
- Si besoin, un grand bloc de code peut être commenté après l'accolade fermante

```
//L'activité dépend du vent
if (weatherIsWindy)
{
    //Nouvelle méthode holistique pour obtenir la force du vent
    //ATTENTION : à valider lors des tests du 12.02
    int windForce = CheckWindForce(now);

    //OK pour catamaran
    if (windForce > MIN_WIND_CATAMARAN && windForce < MAX_WIND_CATAMARAN)
    {
        PrepareStuffForCatamaranActivities();
        SailTheBoat();
        while (NotTooMuchWaves())
        {
            HaveFun();
            ShowKnotSpeed();
            AdjustSailTension();
        }
    }
    //OK pour kitesurf
    else if (windForce > MIN_WIND_KITESURF)
    {
        //Appel d'un webservice externe (attention au délai)
        place = FindASpot();

        //Checkliste du matériel de kite
        PumpTheKite();
        CheckLines();
        CheckSafetySystems();
        LaunchTheKite();

        while (EnoughWind())
        {
            HaveFun();
            TryToStayOnTheBoard();
            TryToDoSomeJumps();

            //TODO récupérer l'exception pour gérer le danger
            BewareOfOtherRiders();
        }
    }
}
//Fin weatherIsWindy
```

Pas utile

Utile

## 3 Règles de codage

### 3.1 Entête

- Chaque fichier doit contenir les informations suivantes (adaptées selon votre contexte)

```

///ETML
///Auteur : Luke Skywalker
///Date : 19.01.2014
///Description : Algorithme qui met à la lumière le côté obscur et le
///              transforme

```

- Chaque **classe**, méthode ou fonction doit avoir les commentaires suivants:

- Description (summary)
- Paramètres (si utile)
- Valeurs de retour (si utile)
- Exceptions levées (si utile)

- Exception

- Si une description est disponible pour une classe, elle peut être absente de l'entête:

```

///ETML
///Author: TranaValentin
///Date: 05.02.2019
using System;

namespace TemplateTableauAProteger
{
    /// <summary>
    /// Template which allows you to move into a table
    /// </summary>
    public abstract class ArrayMoveTemplate
    {

```



En C# avec Visual Studio, vous pouvez générer ceci en écrivant “///” en dessus d’un élément (classe, ou méthode):

```

/// <summary>
/// Convertit une température ddonnée en fahrenheit vers des degrés Celsius
/// Inspiré de http://www.mathsisfun.com/temperature-conversion.html
/// </summary>
/// <param name="temperature">la température en fahrenheit</param>
/// <returns>la température convertie en degré Celsius</returns>
/// <exception cref="Exception">Si la température ne peut être convertie</exception>
public float ConvertToCelsius(float temperature)
{
    const float FARENHEIT_TO_CELSIUS_RATIO = 5 / 9;
    const int FARENHEIT_TO_CELSIUS_DELTA = -32;

    return (temperature + FARENHEIT_TO_CELSIUS_DELTA) * FARENHEIT_TO_CELSIUS_RATIO;
}

```



## 3.2 Nommage

### 3.2.1 Général

Chaque identifiant doit suivre les règles suivantes :

- Être explicite (compréhensible selon son but)
- Écrit en anglais
- Contenir uniquement des caractères standards (pas de #, \*, ^, `, ', ...)
- Les tableaux (ou collections) sont au pluriel.

### 3.2.2 Constante

Les constantes sont écrites en majuscules et séparées par un "souligné".

Exemple : `MAXIMUM_SPEED`, `PI`, `DAYS_IN_A_WEEK`

### 3.2.3 Variable

Le standard est le `lowerCamelCase`<sup>1</sup>.

**C#** : Les attributs (variables de classe) privés sont préfixés par un "\_" (underscore).

**Javascript** : Pour déclarer une variable :

- L'utilisation du mot clé **var** est interdit.
- Il faut privilégier l'utilisation de **const** et quand ce n'est pas possible, il faut utiliser **let**.

Exemple : `aNiceCamel`, `theDarkForce`, `description`, `_privateVar`

### 3.2.4 Les tableaux et les listes

Les tableaux sont des structures de données de type référence. Il regroupe plusieurs données d'un même type, de ce fait son nom sera écrit au pluriel.

Exemples

```
int[] myValues = new int[MAX_VALUE];
```

```
List<string> weekDays = new List<string>() { "Lundi", "Mardi",  
"Mercredi", "Jeudi", "Vendredi", "Samedi", "Dimanche"};
```

### 3.2.5 Méthode

Langage	Règle	Exemple
C#	UpperCamelCase <sup>2</sup>	<code>ConvertToInt32()</code>
PHP, Javascript, Java	lowerCamelCase <sup>1</sup>	<code>convertToInt32()</code>

**Javascript** : Il faut privilégier l'utilisation des fonctions fléchées sauf pour les méthodes de classe

### 3.2.6 Propriété

Langage	Règle	Exemple
C#	UpperCamelCase <sup>2</sup>	<code>SpeedLimit</code>
PHP, Javascript, Java	lowerCamelCase <sup>1</sup>	<code>getSpeed()</code> / <code>setSpeed(5)</code>

<sup>1</sup> lowerCamelCase : minuscules excepté des majuscules pour séparer les mots (premier mot non compris), exemple : `aNiceCamel`

<sup>2</sup> UpperCamelCase : minuscules excepté des majuscules pour séparer les mots (premier mot y compris), exemple : `ANiceCamel`

### 3.2.7 Classe

Les noms de classe suivent le UpperCamelCase<sup>2</sup>.

Exemple : *XWing*, *Trooper*, *Transaction*, *Human*, *GameOfThrone*



































### 3.2.8 Interface (au sens POO)

Langage	Règle	Exemple
C#	Préfixe avec "I"	IContainerControl
PHP, Javascript, Java	Pas de préfixe	ContainerControl

### 3.2.9 Composants graphiques (UI)

Les composants graphiques sont des variables et suivent donc les mêmes règles que ces dernières. Si besoin, elles peuvent être agrémentées avec un mnémonique de type.

Exemple : *exitElement*, *btnExit*, *informationBox*, *titleLabel*, *scoreInput*, *txtCountry*

btn	 Button	lsv	 ListView	rtx	 RichTextBox	spc	 SplitContainer
chk	 CheckBox	mtx	 MaskedTextBox	txt	 TextBox	flp	 FlowLayoutPanel
cbo	 ComboBox	mcd	 MonthCalendar	too	 ToolTip	tbc	 TabControl
dtp	 DateTimePicker	nfi	 NotifyIcon	trv	 TreeView	tlp	 TableLayoutPanel
lbl	 Label	pct	 PictureBox	wbr	 WebBrowser	mnu	 MenuStrip
lkb	 LinkLabel	pgb	 ProgressBar	gpb	 GroupBox	ssp	 StatusStrip
lst	 ListBox	rbt	 RadioButton	pnl	 Panel	tsp	 ToolStrip
tim	 Timer	fbd	 FolderBrowserDialog	ofd	 OpenFileDialog		
cdg	 ColorDialog	fdg	 FontDialog	sfd	 SaveFileDialog		



### 3.2.10 Bases de données

#### 3.2.10.1 Général

- Les bases de données sont préfixées par « db\_ »
- Les tables sont préfixées par « t\_ »

#### 3.2.10.2 Clés

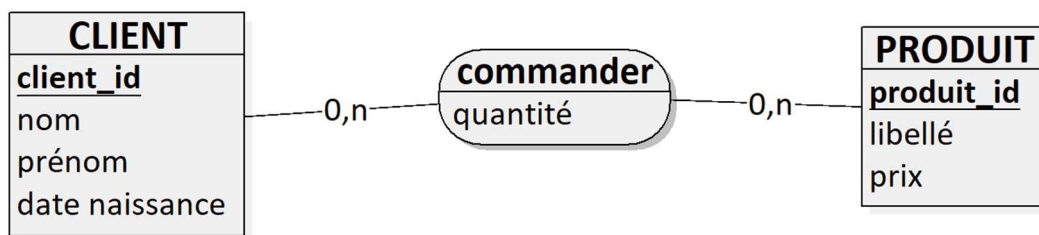
Les clés sont **suffixées** ainsi :

Anglais :      Clé primaire      **product\_id**      Clé étrangère      **product\_fk**

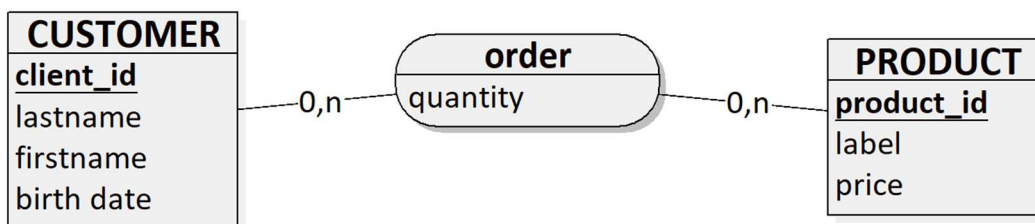
Français :    Clé primaire      **produit\_id**      Clé étrangère      **produit\_fk**

#### 3.2.10.3 Exemples

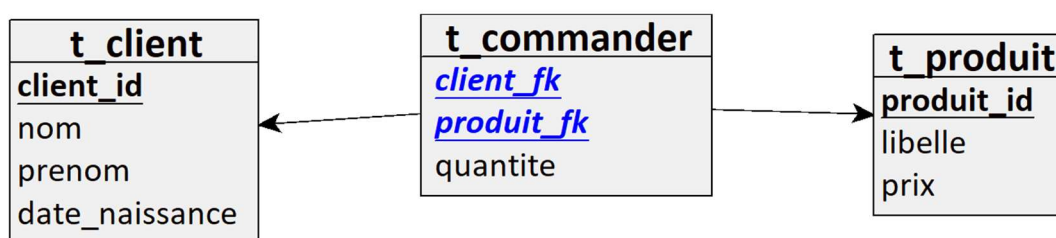
*Exemple MCD en français :*



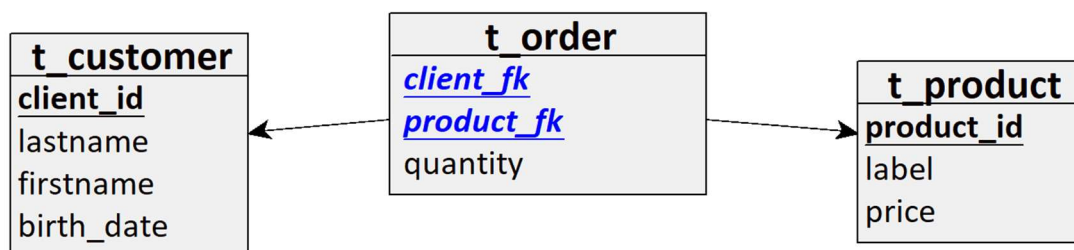
*Exemple MCD en anglais :*



*Exemple MLD en français :*



Exemple MLD en anglais :



#### 3.2.10.4 Outils, Frameworks et jointures naturelles

Si besoin, les normes de base de données peuvent être adaptées à la contrainte technologique.

## 4 Gestion de version

Dans l'idéal, il est recommandé d'utiliser un gestionnaire de version (VCS) comme Git pour historiser les modifications de code et pouvoir facilement travailler en équipe. Dans le cas contraire, il est demandé de noter le suivi de modifications dans l'entête selon l'exemple suivant :

```
///ETML
///Auteur : JMY
///Date : 23.06.2014
///Description : Clone de l'application doodle.ch avec moins de fonctionnalités
///
///Version 2.0.0
///Date : 15.09.2014
///Auteur : CHA
///Description : Ajout du calendrier
///
///Version 2.0.1
///Date : 17.09.2014
///Auteur : AMG
///Description : Bugfix pour les années bissextiles
///
///Version 3.0.0
///Date : 20.09.2015
///Auteur : RFA
///Description : Refactoring complet pour utiliser EntityFramework
```

### 4.1 Numéros de version

Les numéros de version suivent la logique de sémantique 2.0.0 :

- 1<sup>er</sup> digit : version majeure (aucune garantie de compatibilité)
- 2<sup>ème</sup> digit : version mineure (modification légère, compatibilité garantie)
- 3<sup>ème</sup> digit : correction de bug

Référence : <http://semver.org/>

## 5 Apprendre par l'exemple

### 5.1 C#

```
/// <summary>
/// Création de rendez-vous synchronisés
/// </summary>
class Woodle
{
    const string TITLE = "Woodle";

    //Limitation des sondages
    const int MAX_POLLS = 5;
    const int MAX_POLL_CHOICES = 10;

    /// <summary>
    /// Démarrage de l'application
    /// </summary>
    /// <param name="args">arguments passés dans la ligne de commande</param>
    static void Main(string[] args)
    {
        string poll = null;
        string pollUri = null;

        ShowText(TITLE);

        //Demande à l'utilisateur les informations pour le sondage (ctrl-c pour quitter)
        //et affiche l'URI générée
        poll = RegisterPoll();
        pollUri = GenerateURI(poll);
        ShowText("Your poll URI is at http://" + pollUri);

        ShowText("Thanks for using Woodle service, Bye bye");
    }

    /// <summary>
    /// Afficher le texte dans la console
    /// </summary>
    /// <param name="textToShow">Le texte qui va être affiché (\n sera interprété comme nouvelle ligne)</param>
    private void ShowText(String textToShow)
    {
        Console.WriteLine(textToShow);
    }

    /// <summary>
    /// Demande à l'utilisateur les informations du sondage
    /// </summary>
    /// <returns>L'identifiant du sondage généré</returns>
    private string RegisterPoll()
    {
        string[] choices = new string[MAX_POLL_CHOICES];

        //TODO : Implémenter l'enregistrement du sondage

        return "NOT_YET_IMPLEMENTED";
    }
}
```

## 5.2 PHP

```
<?php

/**
 * ETML
 * Auteur: Dark Vador
 * Date: 17.06.2048
 * Description: Utilitaires pour la météo
 */
class WeatherUtils
{
    //Le ratio a été simplifié, 5/9 serait mieux mais compliqué avec une constante PHP
    const FARENHEIT_TO_CELSIUS_RATIO = 0.56;
    const FARENHEIT_TO_CELSIUS_DELTA = -32;

    /**
     * Convertir une température donnée en fahrenheit vers des degrés Celsius
     * @param $temperature => la température en fahrenheit
     * @return la température en Celsius
     */
    public function convertToCelsius(\SplInt $temperature)
    {
        return ($temperature + self::FARENHEIT_TO_CELSIUS_DELTA) * self::FARENHEIT_TO_CELSIUS_RATIO;
    }
}

?>
```

## 5.3 JavaScript

```
/**
 * ETML
 * Auteur : Han Solo
 * Date : 01.02.2028
 * Description : Traite les données envoyées par des sondes sur Naboo
 */

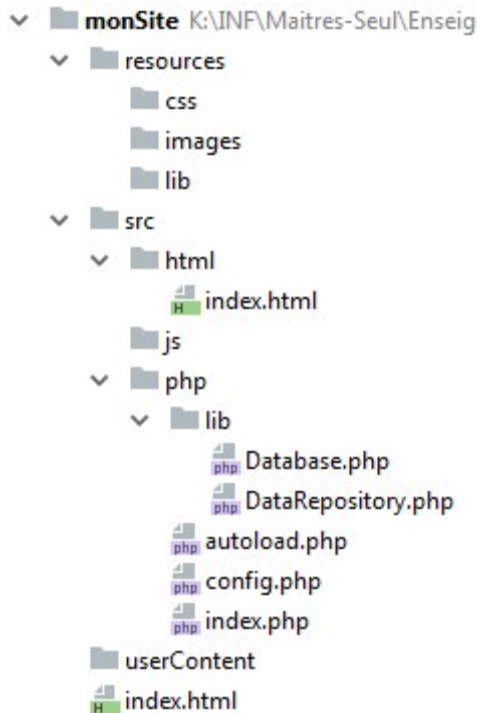
/**
 * Affiche les données des sondes dans l'élément HTML donné
 * @param probes, un tableau de sondes contenant des données
 * @param placeholderId, un emplacement html ou afficher le résultat
 */
function printData(probes, placeholderId) {
    var htmlContent = "";
    var probesSize = probes.length;

    // Préparation des données pour l'affichage
    for (var i = 0; i < probesSize; i++) {
        var probe = probes[i];
        htmlContent += "<tr><td>" + probe.name + "</td><td>" + probe.value + "</td></tr>";
    }

    // Affichage du contenu dans l'élément
    document.getElementById(placeholderId).innerHTML = htmlContent;
}
```

## 5.4 HTML

### 5.4.1 Structure des dossiers



**lib** : utilisé pour les librairies tierces (jquery, prototype.js, mootools,...)

**js** : pour du code JavaScript personnalisé

**lib** : classes et fonctions php du projet

**autoload.php** : pour charger les classes

**config.php** : informations de connexion

**userContent** : données non relatives au design (PDF contrats, photos, éléments uploadés,...). Ce dossier doit être segmenté si besoin.



## 5.4.2 Code

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <!--
    Auteur      : Gabriela Angelo
    Date       : 27.01.2089
    Description : Page index pour le portail de marketing
    -->

    <meta charset="utf-8"/>

    <meta name="author" content="gabriela angelo" />
    <meta name="description" content="marketing homepage" />
    <meta name="keywords" xml:lang="fr" lang="fr" content="marketing, portal" />

    <title>Accueil - Marketing</title>

    <link rel="stylesheet" type="text/css" href="css/portal.css" media="screen" />
</head>
<body>
    <div id="container">
        <header>
            
        </header>

        <section>
            <h1>Contactez-nous</h1>
            <form name="contactForm" class="formGroup">
                <label for="email">Email</label>
                <input id="email" type="text" name="email" />
                <input type="submit" />
            </form>
        </section>

        <footer>
            <a href="mailto:chewbacca@georgelucas.com">email</a>
        </footer>
    </div>
</body>
</html>
```



## 5.5 CSS

```
/**
 * ETML
 * Auteur : Juste Leblanc
 * Date : 21.06.2015
 * Description : Feuille de style pour le layout global du portail marketing
 **/
body {
    margin: 0;
    padding: 0;
    font-size: 62.5%;
}

.formGroup {
    color: blue;
    font-size: 2em; /* Pas encore validé par le client */
}

#container {
    width: 980px; /* Compatibilité IE 6 */
    margin: 0 auto;
    border: 3px solid red;
    background-color: yellow;
}
```