ETML development guidelines

Version 1.1.0

Table of content

1 Scope and purpose

This document is applicable for any coding activity done at ETML which includes :

- Internal and external projects
- Any exercise/drill
- Personal projects realized in a teaching context
- Small scripts/utilities for automation

It's not intended to be used for automated code generation resources or imported code/library.
Nevertheless, if you use some code snippet, the author should be referenced as shown in this example :

```
/// <summary>
/// Convert a temperature given in farenheit into celsius.
/// Inspired from http://www.mathsisfun.com/temperature-conversion.html
/// </summary>
/// <param name="temperature">the temperature in farenheit</param>
/// <returns>the temperature converted into celsius scale</returns>
/// <exception cref="Exception">if temperature cannot be converted</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 Global considerations

2.1 Documentation

As appropriate, an external documentation can be useful and explicitly asked by the teacher/client. The type of information can be varied such as diagrams, interface description or any other relevant element.

Nevertheless, it's crucial to keep a maximum of the information inside the source code itself. This will ensure a better consistency during the entire application lifecycle.

2.2 Algorithm

Algorithms should be motivated by following concerns

- Bug risk reduction
- Easy to upgrade
- Optimized in term of resource use
- Easy to read/understand

2.3 Comments

Commenting code is a good investment for application cost reduction as it helps maintaining the application efficiently. That's why it should help understanding what may not be clear enough or exposed to risky interpretation.

For those reasons it's considered as good practice to **comment group of instructions** instead of commenting each line of code.

Comments should be written in English.

Examples :

Comments

```
/// <summary>
/// This method is useless
/// </summary>
private void uselessComments()
{
    //The minimum power to be a jedi
    const int MIN_POWER_TO_BE_A_JEDI=5;

    //The available balance
    int availablePower = 10;

    //He is not a jedi
    bool isAJedi = false;

    //If I am a jedi and my power is greater than 5 then...
    if (isAJedi && availablePower > MIN_POWER_TO_BE_A_JEDI)
    {
        Console.WriteLine("Congrats, you're a jedi");
    }

}
```

Relevant comments

```
/// <summary>
/// This method is a good example of usefull comments
/// It's not intended to be used in production
/// </summary>
private void usefullComments()
{
    //Validated by the jedi academy on 24 june 2014 (see law #JS121199)
    const int MIN_POWER_TO_BE_A_JEDI = 5;

    //Fixed for testing purpose
    int availablePower = 10;

    //By default, a human is not considered as a jedi (see law #ST4857)
    bool isAJedi = false;

    //This enforces the law of the jedi academy
    //Please beware of adding this check anywhere it may be usefull
    if (isAJedi && availablePower > MIN_POWER_TO_BE_A_JEDI)
    {
        Console.WriteLine("Congrats, you're a jedi");
    }

}
```

2.4 Readability

- The code should be readable, airy and indented
- Constants must be used as much as possible as it helps reducing bugs
- The code should be split in functions (divide to conquer)
- If needed, an big block of code can be commented on the right of the closed bracket :

```
//Activity depends on wind
if (weatherIsWindy)
{
    //Retrieve wind force with holy incantation
    int windForce = checkWindForce(now);

    //Time for catamaran
    if (windForce > MIN_WIND_CATAMARAN && windForce < MAX_WIND_CATAMARAN)
    {
        prepareStuffForCatamaranActivities();
        sailTheBoat();
        while (notTooMuchWaves())
        {
            haveFun();
            showKnotSpeed();
            adjustSailTension();
        }
    }
    //Ready for kitesurfing
    else if (windForce > MIN_WIND_KITESURF)
    {
        //Calls webservice to find a spot nearby
        place = findASpot();

        //Prepare for kiting
        pumpTheKite();
        checkLines();
        checkSafetySystems();
        launchTheKite();

        while (enoughWind())
        {
            haveFun();
            tryToStayOnTheBoard();
            tryToDoSomeJumps();

            //TODO catch exception to handle the danger
            bewareOfOtherRiders();
        }
    }
}//weatherIsWindy
```

unnecessary ✓

relevant ✓

3 Coding rules

3.1 Headers

- Each file should contain the following information

```
///ETML
///Author : Luke Skywalker
///Date   : 19.01.2014
///Summary : Algorithm which transforms the dark force into light...
```

- Each function/method must be documented with following information :
  - Summary
  - Parameters (if relevant)
  - Return values (if relevant)
  - Exceptions thrown (if relevant)

In C# with visualStudio, you can generate the skeleton typing « /// » above the method signature :

```
/// <summary>
/// Convert a temperature given in farenheit into celsius.
/// Inspired from http://www.mathsisfun.com/temperature-conversion.html
/// </summary>
/// <param name="temperature">the temperature in farenheit</param>
/// <returns>the temperature converted into celsius scale</returns>
/// <exception cref="Exception">if temperature cannot be converted</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 Naming

3.2.1 General

Each identifier should follow these rules :

- Must be explicitly self explanatory
- Written in English
- Contains only standard characters (not #, * , ^ , ` , ´ ,_,...)
- Collections (arrays, ...) are plural. A list (array) of country will be named « countries »

3.2.2 Variable

Follow camelcase standard :

- Starts with a lowercase
- Composed parts start with uppercase

Example : aNiceCamel, theDarkForce, description

3.2.3 Constant

Constants are written in uppercase and separated by underscore.

Example : MAXIMUM_SPEED, PI, DAYS_IN_A_WEEK

3.2.4 Method

Follow camelcase standard (like variables) and should start with a infinitive verb.

Examples : flyAway(), checkBackupState(), prepareForTakeOff()

3.2.5 UI components

UI components follow the same rules as variables (as they are variables) and should be self explanatory.

Examples : exitButton, informationBox, titleLabel, scoreRadio, countryCombo

3.2.6 Database

3.2.6.1 General

- Databases are prefixed with « db_ »
- Tables are prefixed with « t_ »

3.2.6.2 MS Access

MS Access additional objects should be prefixed as follow :

f_	form (sf_=subform)
s_	state (ss_=substate)
q_	query (select,insert,update)

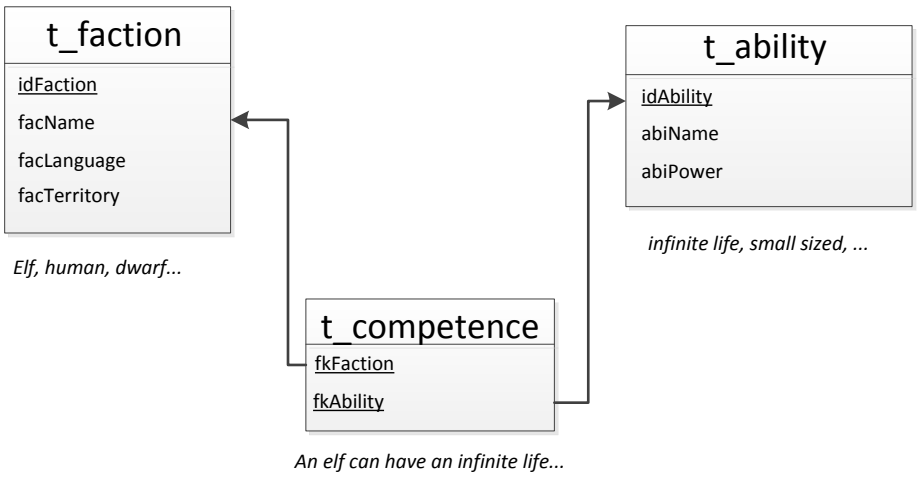
Examples:

db_eyeOfMoon	Database for « eye of moon » project
t_jedi	Table for jedis
f_trainingSession	Form to register training sessions
q_playerStatistics	Query for player statistics

Table keys are prefixed as follow

id	Primary key	fk	Foreign key

Example (MLD) :



3.3 Printing

Code prints should contain line numbers and printed in black and white whenever possible.

4 Learn by example

4.1 C#

```
///ETML
///Author : JMY
///Date : 23.06.2014
///Summary : Clone of doodle.ch application with reduced capabilities

using System;

namespace ch.etml.administration
{
    class Woodle
    {
        const string TITLE = "Woodle";

        //Polls limits
        const int MAX_POLLS = 5;
        const int MAX_POLL_CHOICES = 10;

        //Bootstrap
        static void Main(string[] args)
        {
            string poll = null;
            string pollUri = null;

            showText(TITLE);

            //Ask user for poll informations (ctrl-c to exit) and print generated URI
            poll = registerPoll();
            pollUri = generateURI(poll);
            showText("Your poll URI is at http://" + pollUri);

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

        }

        /// <summary>
        /// Print text to console
        /// </summary>
        /// <param name="textToShow">The text that will be shown (\n will be interpreted as new line)</param>
        private void showText(String textToShow)
        {
            Console.WriteLine(textToShow);
        }

        /// <summary>
        /// Ask user for poll informations
        /// </summary>
        /// <returns>The generated poll internal identifier</returns>
        private string registerPoll()
        {
            string[] choices = new string[MAX_POLL_CHOICES];

            //TODO : implement poll registration

            return "NOT_YET_IMPLEMENTED";
        }
```

4.2 PHP

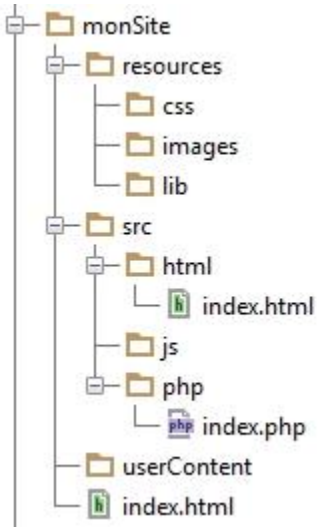
```php
<?php

/**
 * ETML
 * Author: Dark Vador
 * Date: 17.06.2048
 * Summary: Utilities for weather forecasting
 */
class WeatherUtils
{
    // Ratio has been simplified, 5/9 would be better but complicated for PHP constants
    const FARENHEIT_TO_CELSIUS_RATIO = 0.56;
    const FARENHEIT_TO_CELSIUS_DELTA = -32;

    /**
     * Convert temperature from farenheit to celsius
     * @param $temperature => expected temperature in celsius
     * @return the temperature in farenheit
     */
    public function convertToCelsius(\SplInt $temperature)
    {
        return ($temperature + self::FARENHEIT_TO_CELSIUS_DELTA) * self::FARENHEIT_TO_CELSIUS_RATIO;
    }
}

?>
```

4.3 JavaScript

```javascript
/**
 * ETML
 * Author : Han Solo
 * Date : 01.02.2028
 * Summary : Handles data sent by custom probes on the mountain
 */


/**
 * Displays snow probes data in given html placeholder
 * @param probes, an array of probes containing snow data
 * @param placeholderId, an html container in which data will be printed
 */
function printData(probes, placeholderId) {
    var htmlContent = "";
    var probesSize = probes.length;

    // Prepare probe values for display
    for (var i = 0; i < probesSize; i++) {
        var probe = probes[i];
        htmlContent += "<tr><td>" + probe.name + "</td><td>" + probe.value + "</td></tr>";
    }

    // Replace content with probe values
    document.getElementById(placeholderId).innerHTML = htmlContent;
}
```

4.4 HTML

4.4.1 Folder structure



lib : *used for any third party javaScript library (jquery, prototype.js, mootools,...)*

js : *used for custom javaScript code*

userContent : *data not related to layout (pdf contracts, photos, web user uploaded data,...). This folder should be split according to needs.*

4.4.2 Code

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <!--
    Author : Gabriela Angelo
    Date : 27.01.2089
    Summary : Index page of marketing portal
    -->

    <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>Marketing homepage</title>

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

        <section>
            <h1>Contact us</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>
```

4.5 CSS

```css
/**
* ETML
* Author : Juste Leblanc
* Date : 21.06.2015
* Summary : Stylesheet for global layout of the marketing portal
**/
body {
    margin: 0;
    padding: 0;
    font-size:62.5%;
}

.formGroup {
    color: blue;
    font-size:2em; /* Not yet validated by customer */
}

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