|  |  |
| --- | --- |
| Logo_FPT_University_doc |  |

|  |
| --- |
| **FPT UNIVERSITY** |
| Capstone Project Document |

WingS

Coding convention

|  |  |  |
| --- | --- | --- |
| **WingS** | | |
| **Group Members** | Đào Trọng Nghĩa | SE03220 |
| Lê Hồng Nhiên | SE03377 |
| Tạ Ngọc Duy | SE03267 |
| Đỗ Văn Tuấn | SE03297 |
| Hoàng Anh Tuấn | SE03108 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | WS | |

- Hanoi, 10/2016 –

# Code Source

Do not use any libraries or code from external sources without prior approval:

1) Similar code may already exist

2) Licensing may be an issue

3) Legibility and reusability can be issued

4) Performance concerns

# 2. Convention

## 2.1 Case

Use "lowerCamelCase" style (lowercase lettering on initial words and capitalization on subsequent words) to name functions, methods, and variables

## 2.2 Variables

Case should never be used to differentiate between variable names. Every variable name in the current scope should be absolutely unique. Variable names should describe the content that they (will) contain, using either complete words or understandable abbreviations.

## 2.3 Class Definitions

Use the concise syntax to create instances of a delegate type.

// First, in class Program, define the delegate type and a method that

// has a matching signature.

// Define the type.

public delegate void Del(string message);

// Define a method that has a matching signature.

public static void DelMethod(string str)

{

Console.WriteLine("DelMethod argument: {0}", str);

}

## 4 Try-catch

Use a [try-catch](https://msdn.microsoft.com/en-us/library/0yd65esw.aspx) statement for most exception handling.

static string GetValueFromArray(string[] array, int index)

{

try

{

return array[index];

}

catch (System.IndexOutOfRangeException ex)

{

Console.WriteLine("Index is out of range: {0}", index);

throw;

}

}

## 2.5 Function call

Functions should be called with no spaces between the function name, the opening parenthesis, and the first parameter, spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, and the semicolon.

As displayed above, there should be one space on either side of an equals sign used to assign the return value of a function to a variable. In the case of a block of related assignments, more space may be inserted to promote readability.

To support readability, parameters in subsequent calls to the same function/method may be aligned by parameter name.

## 2.8 HTML

HTML Code should not be found anywhere in our code, other than the smarty templates. Mark places where you find it with a TODO, as well as any plain text outputted to screen. We will convert these to a multiple language format in next version.

## 2.9 Control Structures

* Place a single space between the control keyword (if, for, while, switch, etc.) and opening parenthesis to distinguish control statements from function calls
* Always use curly braces—even when technically optional (i.e., avoid PHP's alternative syntax for control structures, except the ternary operator noted below)
* Include break; after all switch case statements
* Use the Allman/BSD style for indentation and layout (braces appear alone and surrounding the indented code)

## 2.10 [&& and || Operators](javascript:void(0))

* To avoid exceptions and increase performance by skipping unnecessary comparisons, use [&&](https://msdn.microsoft.com/en-us/library/2a723cdk.aspx) instead of [&](https://msdn.microsoft.com/en-us/library/sbf85k1c.aspx) and [||](https://msdn.microsoft.com/en-us/library/6373h346.aspx) instead of [|](https://msdn.microsoft.com/en-us/library/kxszd0kx.aspx) when you perform comparisons, as shown in the following example..

Console.Write("Enter a dividend: ");

var dividend = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter a divisor: ");

var divisor = Convert.ToInt32(Console.ReadLine());

// If the divisor is 0, the second clause in the following condition

// causes a run-time error. The && operator short circuits when the

// first expression is false. That is, it does not evaluate the

// second expression. The & operator evaluates both, and causes

// a run-time error when divisor is 0.

if ((divisor != 0) && (dividend / divisor > 0))

{

Console.WriteLine("Quotient: {0}", dividend / divisor);

}

else

{

Console.WriteLine("Attempted division by 0 ends up here.");

}

## 2.11 Data Tables and Fields

* Lowercase words separated by underscores:
* Items
* item\_specifications
* Do NOT make calls directly to native database functions (ie: mysql\_insert\_id, mysql\_result,etc). Use ONLY adodb functions.

## 2.12 Database

* Tables will always be named singular names for most tables, except when dealing with a clear collection. Thus the foreign key relationship will hold true to table\_id form
* Tables will be grouped by prefixes of primary related tables (itemProperties related to table item)
* Additional prefix grouping will be used for tables related to a particular functionality - "workspace", "sys" - this is only because there isn't a parent table, but the tables are in fact related, but by functionality.
* Primary keys of all tables will be lowercase table\_id or write off name table\_id, except in the case of composite keys (those will depend on the combination)
* Foreign keys will be table\_id (so if primary table is "site", and its primary key is "table\_id", when "table\_id" is referenced in other tables, they will reference "site\_id".
* Attribute of table always set name start with name table\_ or write off name table\_ (example :u\_id ( id of user), contact\_id (id of contact)).

# Security

## Password

* Never hardcode passwords into any script
* Never put live data passwords into TortoiseGit
* Always get your database passwords from included csp config.

## Database

* Do not distribute database information. (via email, download or otherwise)
* Do not connect to production server from home (only dev).
* Do not run test or live queries on production data.

## Configuration

* Never hard code paths to any file – use configuration files.

# Code Documentation

**Subversion**

Include comments for all commits to TortoiseGit. If you are fixing a bug, included the TICKET

ID in the Subversion Comments

**Comments**

Use C-style (/\* \*/) comments for multi-line statements and C++ comments (//) for single-line statements.

Use <!-- --> in html mode.