|  |
| --- |
| PS2Win |
| Coding Standards |
| Keep Your Time |

|  |
| --- |
| Rui Ganhoto  07-04-2013 |

Content

[1. Classes 2](#_Toc353618646)

[2. Variable Names 2](#_Toc353618647)

[3. Public Properties 3](#_Toc353618648)

[4. Methods 3](#_Toc353618649)

[5. Arguments 4](#_Toc353618650)

[6. Components 4](#_Toc353618651)

[7. Enumerations 5](#_Toc353618652)

[8. Exception Handling 5](#_Toc353618653)

[9. Comments 6](#_Toc353618654)

**Images**

**No table of figures entries found.**

**Tables**

[Table 1: List of Contribuitors 2](#_Toc353618655)

[Table 2: Version history 2](#_Toc353618656)

[Table 3: Variable prefix and Initialization Values 3](#_Toc353618657)

[Table 4: Component Naming Prefix 5](#_Toc353618658)

|  |  |  |  |
| --- | --- | --- | --- |
| **Authors and Contributors** | | | |
| **Date** | **Name** | **Contacts** | **Contribution** |
| 07-04-2013 | Rui Ganhoto | a21170262@alunos.isec.pt | Author |
| 11-04-2013 | Mário Oliveira | a21170292@alunos.isec.pt | Contributor |
|  |  |  |  |
|  |  |  |  |

Table : List of Contribuitors

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Revision History** | | | | | |
| **Date** | **Description** | **Author** | **Version** | **Approvers** | **State** |
| 07-04-2013 | Creation of first draft | Rui Ganhoto | 0.1 |  | Draft |
| 07-04-2013 | Ready for Revision | Rui Ganhoto | 0.2 |  | Ready for Revision |
| 11-04-2013 | Document reviewed | Mário Oliveira | 0.2 |  | Ready for Revision |
| 13-04-2013 | Document Correction | Rui Ganhoto | 0.3 |  | Ready for Revision |
| 15-04-2013 | Document reviewed | Mário Oliveira | 0.3 |  | Ready for Revision |

Table 2: Version history

# 1. Classes

Classes should have the first character of all words is upper case and the others are lower case. Classes must be complete names without abbreviations.

A Class name must start with a Letter and only contains Letters and Numbers, any other chars will not be allowed and will give origin to a technical unconformity.

Classes will have an XML Comment with the summary of the class.

**Example:**

/// <summary>

///Contains Message Methods and Events

/// </summary>

Public class Messages { }

# 2. Variable Names

Variables should be first character of all words is upper case and the others are lower case, except for the first word that is lower case.

A Variable name must start with the prefix and then using letters and numbers only, exceptionally underscore (“\_”) will be used.

Variables should begin by identifying its type using the notation on the table.

Variables should be initialized with mentioned default values, if otherwise, the value must be well documented (follow comments section).

|  |  |  |
| --- | --- | --- |
| Type | Prefix | Initialization Values |
| Int | int | -1; 0 |
| Bool | bln | false |
| String | str | “”; String.Empty |
| Float | flt | 0; |
| Char | chr | ‘’ |
| Double | dbl | 0; |
|  |  |  |

Table 3: Variable prefix and Initialization Values

**Example:**

bool blnCanDoSomething = false;

private intSecondsCounter = 0;

# 3. Public Properties

Properties should be first character of all words is upper case and the others are lower case. Properties must be complete names without abbreviations.

**Example:**

public int NumberOfTasks {get;set;}

# 4. Methods

Method Names should have the first character of all words is upper case and the others are lower case. Methods must be complete names without abbreviations.

Methods must not contain any special Characters, only letters and numbers are allowed.

Methods should be simple and small doing only a simple task. If needed a method should have comments to divide sections on that method.

Every method must have XML header with a small description of the method and arguments, GhostDoc will be used to help creating XML headers.

**Example:**

/// <summary>

///Sends a Message

/// </summary>

void SendMessage() { }

# 5. Arguments

Arguments should have the first character of all words is upper case and the others are lower case. Arguments must be complete names without abbreviations.

Every method must have XML header with a small description of the method and arguments, GhostDoc will be used to help creating XML headers.

**Example:**

/// <summary>

///Sends a Message

/// </summary>

/// <param name=" Message">Message to be sent</param>

void SendMessage(string Message) { }

# 6. Components

Components should be first character of all words is upper case and the others are lower case, except for the first word that is lower case.

Components should begin by identifying its type using the prefix on the table.

|  |  |
| --- | --- |
| Component | Prefix |
| Textbox | txt |
| Label | lbl |
| Image | img |
| CheckBox | chk |
| Spin | spn |
| Panel | pnl |
| SplitContainer | spc |
| Button | btn |
| GroupBox | grp |
|  |  |
|  |  |

Table 4: Component Naming Prefix

**Example:**

chkShowResults

# 7. Enumerations

Enumerations must be used like Classes (for enumeration type) and Properties (for values).

At least, the first value on the enumeration must be set as a constant, this will reduce errors

**Example:**

/// <summary>

/// Enumeration for Short Week Day Name

/// </summary>

enum Days {Sat=0, Sun, Mon, Tue, Wed, Thu, Fri};

# 8. Exception Handling

Exception handling is mandatory on every method.

Exception messages should be readable and friendly to the user and a complete exception report should be provided so the user can send the exception.

A log file can be recorded with any exceptions.

To handle exceptions a class will be created that easily manage exceptions and current method status.

The class will contain:

* Exception Message
* Status {Success; Cancel; Exception; …}
* Stack Trace
* Query
* Affected Lines (used for SQL queries)
* (Any other needed values)

If there is any method that returns a status that differs from Success a dialog or any other type of message will be shown to the user to inform about the error found.

# 9. Comments

A simple code will require very few comments.

Comments will be mandatory in this cases:

1. XML comment on Classes and Methods
2. If any complex or weird logic is needed (What? Or Why?)
   * Ex.: //this section is converting seconds to hours, minutes and seconds (Answering to “What does this section?” or “Why am I doing this section?”)
3. If any initialization differs from default initialization values (Why?)
   * Ex.: int intHourCountdownInSeconds = 3600; //There are 3600 seconds in one hour (Answering to “why am I using this initialization?”)
4. If any weird variable name is declared or not understandable at first sight (What?)
   * Ex.: object aux; //Auxiliary value for multiple visual objects (Answering to “What is this variable?”)