

Coding Conventions

C#, Onix Work

Coding Conventions C#, Onix Work

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| No | Date | Content | Author |
| 1 | 30.10.2018 | Create first version | ld |
|  |  |  |  |

# Approvals

|  |  |  |  |
| --- | --- | --- | --- |
| No | Date | Role | Approved By |
|  |  |  |  |

# **Table of Contents**

C#, Onix Work

[Revision History 1](#_Toc528675074)

[Approvals 1](#_Toc528675075)

[**Table of Contents** 2](#_Toc528675076)

[**Table of Figures** 2](#_Toc528675077)

[Introduction 3](#_Toc528675078)

[Purpose 3](#_Toc528675079)

[Related Documents 3](#_Toc528675080)

[C#, Onix Work 4](#_Toc528675081)

[**Naming Conventions** 4](#_Toc528675082)

[**Indentation** 8](#_Toc528675083)

[**Layout** 8](#_Toc528675084)

[**Exception Handling / Logging** 9](#_Toc528675085)

[**Comment** 10](#_Toc528675086)

# **Table of Figures**

No table of figures entries found.

# Introduction

## Purpose

The purpose of this document is describing about our code conventions

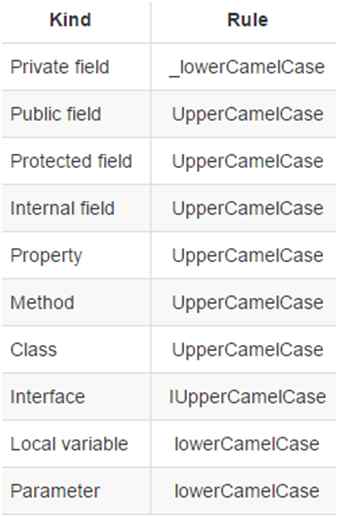
## Related Documents

-

# **C#, Onix Work**

### Naming Conventions

**Terminology**

* **Camel Case (camelCase):** In this the first letter of word always in small letter and after that each word with capital letter.
* **Pascal Case (PascalCase):** In this the first letter of every word is in capital letter.
* **Underscore Prefix (\_underScore):** For underscore ( \_\_ ), the word after \_ use camelCase terminology.  
    
  

**Native DataType**  
Always use native datatype instead of .NET CTS type. For example, use **int** instead of **Int32** or **Int64**.

1. //Good
2. **private** **int** \_salary = 100;
4. //Bad
5. **private** Int16 \_salary = 100;
6. **private** Int32 \_salary=100;

**Class**  
Always use **PascalCase** for class names. Try to use noun or noun phrase for class name.

1. **public** partial **class** About : Page
2. {
3. //...
4. }

**Methods**  
Always use PascalCase for method names. Use maximum 7 parameter for a method.

1. **public** **string** GetPosts(**string** postId)
2. {
3. //...
4. }

**Note**: Don't use name as all character in CAPS.  
 **Arguments and Local Variable**  
Always use **camelCase** with method arguments and local variables. Don't use Hungarian notation for variables.

1. **public** **string** GetPosts(**string** postId
2. {
3. **int** numberOfPost = 0;
4. }

**Note:**Don't use abbreviations for any words and don't use underscore ( \_ ) in between any name.  
  
**Property**  
Use **PascalCase** for property. Never use Get and Set as prefix with property name.

1. **private** **int** \_salary = 100;
2. **public** **int** Salary
3. {
4. **get**
5. {
6. **return** \_salary;
7. }
8. **set**
9. {
10. \_salary = value;
11. }
12. }

**Note:** Don't use name with start with numeric character.  
  
**Interface**  
Always use letter "**I**" as prefix with name of interface. After letter I, use PascalCase.

1. **public** **interface** IUser
2. {
3. /// <summary>
4. /// Check user is exists or not
5. /// </summary>
6. /// <returns>return bool value</returns>
7. **bool** ValidateUser();
8. }

**Private Member Variable**  
  
Always try to use **camelCase** terminology prefix with underscore ( \_ ).

1. **private** **int** \_salary = 100;

**Public Member Variable**  
Always use **PascalCase** for public member variable,

1. **public** **int** Salary = 100;

**Member variable**  
Declare member variable at the **top** of the class, If class has **static** member then it will come at the **top most**and after that other member variable.

1. **public** **class** Account
2. {
3. **public** **static** **string** BankName;
4. **public** **static** **decimal** Reserves;
5. **public** **string** Number
6. {
7. **get**;
8. **set**;
9. }
10. **public** DateTime DateOpened
11. {
12. **get**;
13. **set**;
14. }
15. **public** DateTime DateClosed
16. {
17. **get**;
18. **set**;
19. }
20. **public** **decimal** Balance
21. {
22. **get**;
23. **set**;
24. }
25. // Constructor
26. **public** Account()
27. {
28. // ...
29. }
30. }

**Enum**  
Always use**singular noun** to define enum.

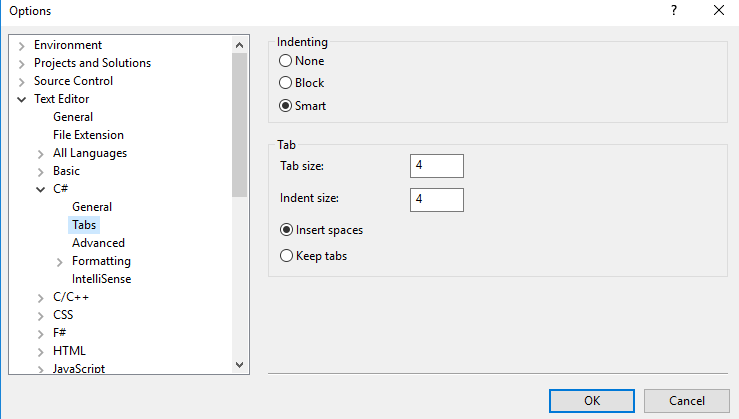
1. **enum** MailType
2. {
3. Html,
4. PlainText,
5. Attachment
6. }

**Namespace**  
Always use **PascalCase** for namespace.  
  
*namespace NextProgramming.Domain*

### Indentation

Use four (4) space indents for blocks

Use four (4) for tabs



### Layout

Braces ALWAYS go on their own line:

class MyClass {

int FuncDoSomeThing()

{

if (something)

{

// ...

}

else

{

// ...

}

}

}

We require braces around the statements for a conditional.

if (condition)

{

body();

}

### Exception Handling / Logging

Don't ignore exceptions, such as:

void SetServerPort(String value) {

try {

serverPort = Integer.parseInt(value);

} catch (NumberFormatException e) { }

}

Use try/catch/finally blocks around code that can potentially generate an exception.

In catch blocks, always order exceptions from the most specific to the least specific.

Use a finally block to clean up resources, whether you can recover or not.

internal class Program

{

private static void Main()

{

long result = 0;

try

{

result = Fibonacci(int.MaxValue);

}

catch (StackOverflowException)

{

// Never execute.

Console.WriteLine("Inside catch.");

}

finally

{

// Never execute.

Console.WriteLine("Inside finally.");

}

// Never execute.

Console.WriteLine(result);

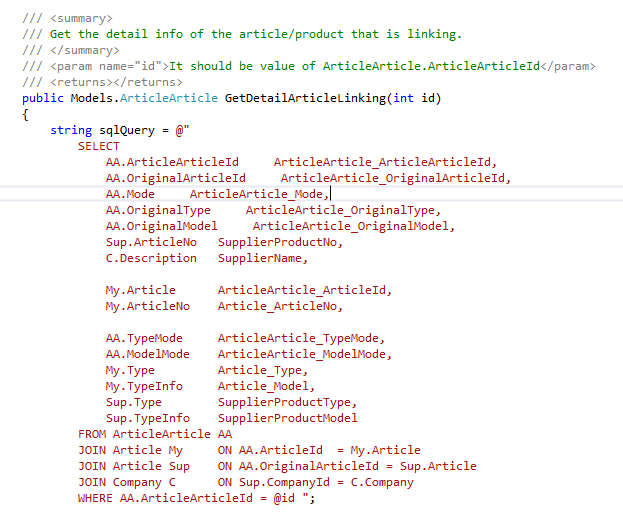
}

}

### Comment

At the first time create new function, MUST create the standard comment on the top of the function explain its purpose and meaning/usage of parameters if there is any.

Whenever we change the return function or list of parameters, we also need to update the top comment to reflect the new function



If we only add more code or change existing code of the function, we MUST add one more comment line to the top comment as below:

Therefor whenever we change the code, we will have things to verify to make sure that it will not cause any troubles for previous bugs/features

