OPD_Bare Necessities - .Net

Document Details

Document Path	http://192.168.1.102:9073/Mindshare/5gpal/policies/OPD_Bare Necessities
Document Version	1.1
Document Date	November 15, 2008
Document Status	Final
Circulation Type	Internal
Circulation List	All@ 5G

Revision History

Ver No. & Date	Added/ Revised By	Content Added/ Changed	Reviewed/ Approved By	App. Date	Broadcast Date	Effective Date
1.0/Apr 2008	Shogan	Base Document	Ananth RK	Apr 2008	Apr 2008	Apr 2008
1.1 / Nov 15, 2008	5G PG Team	Adopted new naming conventions, Added sections on engineering documents etc,	Sumitra Seshan	Nov 15, 2008	Jan 15, 2009	Jan 15, 2009

Table of Contents

1. Introduction

1.1 *Purpose*

Standards defined here must be followed when writing coding.

1.2 *Audience*

Developers and code reviewers.

2. Naming Conventions

2.1 *General*

1. Use abbreviations sparingly.
2. Use descriptive names.
3. Unless explicitly stated do not use underscores while naming.

2.2 *Namespace Naming Guidelines*

CompanyName.TechnologyName[.Feature][.Design]
E.g. FifthGen.BMS.Core, FifthGen.BMS.Finance, FifthGen.BMS.Finance.GL

Also, use plural namespace names if it is semantically appropriate. For example, use System.Collections rather than System.Collection. Exceptions to this rule are brand names and abbreviations. For example, use System.IO rather than System.IOs.

Do not use the same name for a namespace and a class. For example, do not provide both a Debug namespace and a Debug class.

2.3 *Class Naming Guidelines*

1. Use a noun or noun phrase to name a class.
2. Use Pascal Case.
   The following are examples of correctly named classes: FileStream, Button, String

2.4 *Interface Naming Guidelines*

1. Name interfaces with nouns or noun phrases, or adjectives that describe behavior. For example, the interface name IComponent uses a descriptive noun. The interface name ICustomAttributeProvider uses a noun phrase. The name IPersistable uses an adjective.
2. Use Pascal Case.
3. Prefix interface names with the letter I, to indicate that the type is an interface.
4. Use similar names when you define a class/interface pair where the class is a standard implementation of the interface. The names should differ only by the letter I prefix on the interface name.

The following are examples of correctly named interfaces. IServiceProvider, IFormatable

2.5 *File Naming Guidelines*

A file name should reflect the class or interface it contains.

2.6 *Enumeration Naming Guidelines*

- Use Pascal Case for Enum types and value names.
- Do not use an Enum suffix on Enum type names.

E.g.

```
public enum EmployeeType
{
    Permanent = 1,
    Temporary = 2,
    Consultent = 3,
    Trainee = 4
}
```

2.7 *Parameter Naming Guidelines*

1. Use camel Case for parameter names.
2. Use names that describe a parameter's meaning rather than names that describe a parameter's type. Development tools should provide meaningful information about a parameter's type. Therefore, a parameter's name can be put to better use by describing meaning.
3. Do not prefix parameter names with their types.

The following are examples of correctly named parameters.

Type GetType(string typeName), string Format(string format, object[] args)

2.8 *Method Naming Guidelines*

- Use verbs or verb phrases to name methods.
- Use Pascal Case.

The following are examples of correctly named methods. RemoveAll(), GetCharArray(), Invoke() E.g.

```
void ReadFromFile(string fileName)
{
    try
    {
        // read from file
    }
    catch (FileIOException ex)
    {
        // log error

        //  re-throw exception depending on your case
        throw;
    }
}
```

2.9 *Property Naming Guidelines*

- Use a noun or noun phrase to name properties.
- Use Pascal Case.

2.10 *Event Naming Guidelines*

1. Use Pascal Case.
2. Use an EventHandler suffix on event handler names.
3. Specify two parameters named sender and e. The sender parameter represents the object that raised the event. The sender parameter is always of type object, even if it is possible to use a more specific type. The state associated with the event is encapsulated in an instance of an event class named e. Use an appropriate and specific event class for the e parameter type.
4. Name an event argument class with the EventArgs suffix.
5. Consider naming events with a verb. For example, correctly named event names include Clicked, Painting, and DroppedDown.
6. Use a gerund (the "ing" form of a verb) to create an event name that expresses the concept of pre-event, and a past-tense verb to represent post-event. For example, a

Close event that can be canceled should have a Closing event and a Closed event. Do not use the BeforeXxx/AfterXxx naming pattern.

7. Do not use a prefix or suffix on the event declaration on the type. For example, use Close instead of OnClose.
8. In general, you should provide a protected method called OnXxx on types with events that can be overridden in a derived class. This method should only have the event parameter e, because the sender is always the instance of the type.

The following example illustrates an event handler with an appropriate name and parameters.

```
public delegate void MouseEventHandler(object sender, MouseEventArgs e);
```

The following example illustrates a correctly named event argument class.

```
public class MouseEventArgs : EventArgs
{
   int x;
   int y;

   public int X
   {
      get { return x; }
   }

   public int Y
   {
      get { return y; }
   }

   public MouseEventArgs(int x, int y)
   {
      this.x = x; this.y = y;
   }
}
```

2.11 *Variable Naming Guidelines*

- Use camel Case.
- Prefix a non-local variable with an underscore (_).
- Do not prefix/suffix a variable name with its type.

2.12 *Constants*

- Use capital letters while naming constants.

  E.g. private const MAXCOUNT = 100;

2.13 *Control Naming Guidelines*

- Use camel Case.
- Prefix the control name with control type.

Possible prefixes are listed below:

Control	Prefix	Example
Button	btn	**btnOK**
Check Box	chk	**chkAgreed**
Combo Box	cbo	**cboCountry**
List Box	lst	**lstCountry**
Frame	frm	**frmRight**
Group Box	grp	**grpAddressDetails**
Radio Button	rdo	**rdoCheckAll**
Text Box	txt	**txtCustomerName**
Label	lbl	**lblCustomerName**
List View	lvw	**lvwCustomerList**
Tree View	tvw	**tvwOrganizationStructure**
Data Grid	dgr	**dgCustomerList**
Image List	img	**imgToolBar**
Tool Bar	tbar	**tbarTop**
Status Bar	sbar	**sbarMain**
Panel	pnl	**PnlMain**
Menu	mnu	**mnuMain**
Menu Item	mnuItem	**mnuItemNew**

2.14 *Summary*

The following table summarizes the different capitalization styles.

Identifier	Case	Example
Namespace	Pascal	**FifthGen.BMS.Finance**
Class	Pascal	**Account**
Interface	Pascal	**IDispose**
Enumerations	Pascal	**AccountType**
Property	Pascal	**BackColor**
Method	Pascal	**GetAllAccounts()**
Parameters	Camel	**accountNumber**
Variables	Camel	**accountName**
Events	Pascal	**Clicked**
Controls	**Camel**	**txtAccountName**

3. Commenting the code

3.1 *File Comments*

None

3.2 *Class Comments*

At the top of every class provide a description as follows:

```
/// <summary>
/// Purpose of the class
/// </summary>
/// <remarks>
/// Detailed description of the class, if available.
/// </remarks>
```

3.3 *Method Comments*

At the top of every method / function provide a description as follows:

```
/// <summary>
/// Purpose of the method goes here
/// </summary>
/// <param name="paramName1">description of param 1</param>
/// <param name="paramName2">description of param 2</param>
/// <exception cref="ArgumentException">
/// when this exception will be thrown.
/// </exception>
/// <exception cref="NullException">
/// when this exception will be thrown.
/// </exception>
/// <remarks>special remarks, limitations, permissions, etc. </remarks>
```

3.4 *Variable Comments*

For commenting variables, we classify them into two categories:

Member variables

Non-member variables (e.g. local variables)

Member variables are commented as follows:

```
/// <summary>
/// description
/// </summary>
string _employeeName = string.Empty;
```

Non-member variables are commented as follows:

```
string nothing = string.Empty; //just for an example
```

4. Coding Style

4.1 *Line Spacing*

1. Two blank lines should always be used between:
    - Logical sections of a source file (e.g. between declarations and start of methods)
    - Class and interface definitions
2. One blank line should always be used between:
    - Methods
    - Properties
    - Local variables in the method and its first statement
    - Logical sections inside the code to improve readability
    - System namespaces and custom namespaces.

4.2 *Inter-term Spacing*

- There should be a single space after a comma or semicolon.
  E.g. MethodName(a, b, c); (and NOT MethodName(a,b,c);)
- Single spaces should surround operators (except unary operators like ++, --, etc.)
  E.g.
  a = b; (and NOT a=b;)
  for (int I = 0; I < 10; I++) (and NOT (for int I=0;I<10;I++);)

4.3 *Indentation*

Use three space, tabbed indentation.

4.4 *Wrapping Lines*

If the line goes beyond the visual part of the editor, wrap the lines appropriately.

4.5 *Code Regions*

- Member Variables – Group all member variable declarations under region "Variable Declarations"
- Interface implementations – group the entire interface implementations under a region named by the interface.
- Event Handlers – group entire event handlers under region "Event Handlers".
- Inside "Event Handlers" region group sub-regions for each type of event handlers. E.g. "Form Event Handlers", "Menu Event Handlers", "Button Event Handlers", etc. Group each event handler by a region of its own name.
- Methods – Group all methods under a region named "Methods". Group each method by a region of its own name.

5. Database Guidelines

5.1 *Table Naming Guidelines*

- Use Pascal Case.
- Do not use abbreviations.
- Do not prefix/suffix anything to table names.
- Unless you have a strong reason name tables in plural.

5.2 *Field Naming Guidelines*

- Use Pascal Case.
- Do not use abbreviations.
- Do not prefix the field name with data types.

5.3 *Stored Procedure Naming Guidelines*

- Use Pascal Case.
- Refer Method naming guidelines.
- Prefix the procedure name with usp_.

5.4 *View Naming Guidelines*

- Use Pascal Case
- Refer Method naming guidelines.
- Prefix the view name with vw_.

- End of Document -