项目文件组织及代码开发规范

目　录

项目文件组织及代码开发规范 1
1、代码规范的重要性 2
2、开发代码命名规范介绍 2
3、命名总体规则 2
4、项目及文件夹命名规范 2
4.1 类命名规范： 2
4.2 类、方法所属字段（变量）命名规范 3
4.3 方法规范 4
4.4 属性规范 5
4.5 参数规范 5
4.6 常量规范 5
4.7 接口规范 6
4.8 命名空间规范 6
5、代码编写规范 6
5.1 缩进规则 6
5.2 比较规则 6
5.3 对齐规则 7
5.4 模块化规则 7
5.5 模块设计原则 7
5.6 数据库操作 7
5.7 函数返回值原则 8
5.8 注释风格规范 8
5.9 文件规范 8
6、常见语句书写规则 8
7、常见数据类型缩写表 10
8、服务器控件名称缩写表(MVC 基本上不会出现这些控件) 11
8.1 web 控件名称缩写表 11
8.2 html 控件名称缩写表 13

1、代码规范的重要性

1) 规范的代码可以促进团队合作；
2) 规范的代码可以减少 bug 处理，查找 bug 也变得轻而易举；
3) 规范的代码可以降低维护成本；
4) 规范的代码有助于代码审查；
5) 养成代码规范的习惯，有助于程序员自身的成长。

2、开发代码命名规范介绍

1) Pascal 规范：大小写形式－所有单词第一个字母大写，其他字母小写。
2) Camel 规范：大小写形式－除了第一个单词，所有单词第一个字母大写，其他字母小写。

3、命名总体规则

1) 命名中一律使用英文单词，不能使用汉语拼音。
2) 所有项目、文件夹、代码文件都要有与本身实现功能相对应的名称，要求清晰明了，无歧义。
3) 名字尽量不使用缩写，除非它是众所周知的。
4) 名字可以有两个或三个单词组成，但不应多于三个，控制在 3 至 30 个字母以内。
5) 名字尽量使用前缀而不是后缀。

4、项目及文件夹命名规范

1) 项目及存放代码文件的文件夹，使用 Pascal 命名规则命名：首字母大写。
2) 存放资源的文件夹可使用小写，如： css、img、js、dll 等。

4.1 类命名规范：

A. 使用 Pascal 规则命名类名，首字符要大写；
B. 使用能够反映类功能的名词或名词短语命名类；
C. 类文件的名称要能反映类的内容。

4.2 类、方法所属字段（变量）命名规范

A. 使用 camel 规则命名类成员字段；

B. 字段名称= 变量的前缀 +代表变量含意的英文单词或单词缩写；

C. 类变量前缀为：“m_” +数据类型缩写（小写）（其中，m 为“memory”缩写，数据类型缩写见 7.1 常见数据类型缩写表）。如：

```
public class Hello
{
        private string m_strName;
        private DateTime m_dtDate;
}
```

D. 类的属性所对应的变量，采用属性名前加“m_”+ 类型缩写 前缀的形式。如：

```
public class Hello
{
    private string m_strName;
    public string Name
    {
        get
        {
        return m_strName;
        }
    }
}
```

E. 函数级的变量使用”_”+类型缩写前缀。如：

```
public class Hello
{
    void say()
    {
        string _strSayWord;
    }
}
```

F. **补充说明：**

a) 针对异常捕获过程中的 Exception 变量命名，在没有冲突的情况下，统一命名为 exp；

b) 如果有冲突的情况下，可以用“exp”+ 标志名称，如：expSql。

```
Try
{
    //your code
    try
    {
        //code
    }
    catch(Exception exp)
    {
        //your code
    }
}
catch(Exception expSql)
{
//your code
}
```

c) 如果捕获异常不需要作任何处理，则不需要定义 Exception 实例。如：

```
try
{
    //your code
}
catch( Exception exp)
{
}
```

d) 对于可能仅出现在几个代码行中的生存期很短的变量，仍然要求使用有意义的名称。仅对于短循环索引使用单字母变量名，如 i 或 j。

e) 不要使用原义数字或原义字符串，如 For (i = 1;i <= 7;i++)。而是使用命名常数，如 For (i = 1;i <= NUM_DAYS_IN_WEEK;i++) 以便于维护和理解。

4.3 方法规范

A. 方法名使用 Pascal 规则，首字符要大写。

B. 方法名应使用动词或动词短语。

C. 在方法声明前必须用以下格式编写注释

```
/// <summary>
/// depiction:<对该方法的说明>
/// </summary>
/// <param name="<参数名称>"><参数说明></param>
/// <returns>
///<对方法返回值的说明，该说明必须明确说明返回的值代表什么含义>
/// </returns>
///Writer：作者中文名
///Create Date:<方法创建日期，格式：YYYY-MM-DD>
```

4.4 属性规范

A. 属性使用 Pascal 规则，首字符大写。

B. 属性名称和相应字段名称要关联，可以使用“封装”命令来生成属性。

4.5 参数规范

A. 参数采用 camel 规则命名，首字符小写；

B. 使用描述性参数名称，参数名称要具有很强的说明性；

C. 参数使用“p_”+类型缩写作为前缀（其中，p 为“parameter”缩写）。如：

```
public class Hello
{
    void say(string p_strSayWord)
    {
    }
}
```

4.6 常量规范

只读常量、枚举名、静态字段或属性名等均使用 Pascal 规则，使用前缀(对应于常量类型,分别为:const_、enum_、static_)，并在各字母间用下划线分隔。如：

```
private const bool  const_Web_EnablePageCache_Default = true;
private const int  const_Web _PageCacheExpireTime_ Default = 3600;
```

```
private const bool  const_Web_EnableSSL = false;
```

4.7 接口规范

A. 接口定义使用 Pascal 规则，且必须以大写“I”开头；

B. 接口名称要有意义，中间不要有下划线“_”等字符；

C. 如果类实现了接口，名称尽量和接口相同，只是省掉“I”字符。

4.8 命名空间规范

A. 命名空间名称采用 Pascal 规则，首字符大写；

B. 命名空间名称尽量反映其内容所提供的整体功能。

5、代码编写规范

5.1 缩进规则

A. 使用一个“Tab”为每层次缩进。

B. 同一层次的“{}“应对齐，即需要在 VS 提供格式的基础上对尾“}”进行位置调整。例如：

```
function Func()
{
   if (something bad)
   {
        if (another thing bad)
        {
            while (more input)
            {
            }
        }
    }
}
```

5.2 比较规则

总是将恒量放在等号/不等号的左边。一个原因是假如你在等式中

漏了一个等号，语法检查器会为你报错。第二个原因是你能立刻找到数值而不是在你的表达式的末端找到它。例如：

```
if ( 6 == $errorNum ) ...
```

5.3 对齐规则

变量的申明和初始化都应对齐。例如：

```
Int   m_iCount; //注释
Int   i,j;      //注释
Float m_fIncome,m_fPay; //注释


m_iCount = 0;
I = 1;
m_fIncome = 0.3;
```

5.4 模块化规则

某一功能，如果重复实现一遍以上，即应考虑模块化，将它改写成通用函数,向小组成员发布并在内部博客(http://192.168.1.24)进行更新说明。同时要尽可能利用其它人的现成模块。

5.5 模块设计原则

A. 函数要功能单一，不允许一个函数实现两个及两个以上的功能。
B. 不能在函数内部使用全局变量，如要使用全局变量，应先转化为局部变量。
C. 函数与函数之间只允许存在包含关系，而不允许存在交叉关系。即两者之间只存在单方向的调用与被调用，不存在双向的调用与被调用。

5.6 数据库操作

A. 从数据库表或视图提取数据时，只能取出必需的那些字段。

B. 所有关于数据库的操作都通过存储过程来实现，并由存储过程承担功能模块内简单业务逻辑的识别、处理工作。

C. 如需进行并发控制、事务控制等功能，均要求放在存储过程内实现。

D. 清楚明白地使用列名，而不能使用列的序号。

E. 详情可见《数据库各对象命名及使用规范》。

5.7 函数返回值原则

A. 避免使用结构体等复杂类型。

B. 使用 bool 类型：该函数只需要获得成功或者失败的返回信息时候。

C. 使用 int 类型：错误代码用负数表示，成功返回 0。

5.8 注释风格规范

A. 注释应该准确、简洁、有重点。

B. 应该写优雅的、可读性良好的代码，而不是用晦涩的代码写注释。

C. 原则上应尽量减少程序体内代码的注释，应该保持代码本身的直接可读性。

D. 函数的注释，可以只对 public 或者重要的 private 函数进行注解。

5.9 文件规范

A. 避免使用大文件。如果一个文件里的代码超过 300～400 行，应该考虑将代码分开到不同类中。

B. 避免写太长的方法。一个典型的方法代码在 1～30 行之间。

C. 如果一个方法发代码超过 50 行，应该考虑将其分解为不同的方法。

6、常见语句书写规则

如下表所示：

语句	规范风格

if	If (condition)
	{
	　　statements;
	}
	else
	{
	　　statements;

	}

for	for(initialization; condition; update)
	{
	statements;
	}

foreach	foreach(something in collection)
	{
	statements;
	}

switch	switch(…)
	{
	case ..:
	break;
	case …:
	break;
	default:
	}

while	while(..)
	{
	statements;
	}

do-while	do
	{
	statements;
	}
	while(condition);

try-catch	try
	{
	statements;
	}
	catch(Exception exp)
	{
	handle exception;
	}

同一代码块内的不同逻辑块之间应空一行	{
	do statement1;
	
	do statement2;
	}

函数与函数之间至少空一行，但不超三行	

7、常见数据类型缩写表

数据类型	缩写
string	str
int	i
char	sz
sbyte	sb
byte	bt
uint	ui
long	l
ulong	ul
float	f
double	d
bool	b
decimal	dec

8、服务器控件名称缩写表(MVC 基本上不会出现这些控件)

8.1 web 控件名称缩写表

web 控件名	缩写
AdRotator	art
Button	btn
Calendar	cd
CheckBox	chk
CheckBoxList	chkl
CompareValidator	cpv
CustomValidator	ctv
DataGrid	dg
DataList	dl
DropDownList	ddl
HyperLink	hl
Image	img
ImageButton	Icmd

Label	lab
LinkButton	lbtn
ListBox	lst
Panel	pl
PlaceHolder	ph
RadioButton	rb
RadioButtonList	rbl
RangeValidator	rv
RegularExpressionValidator	rev
Repeater	rp
RequiredFieldValidator	rfv
Table	tb
TableCell	tc
TableRow	tr
TextBox	txt
ValidationSummary	vs

XML	XML

8.2 html 控件名称缩写表

html 控件名	缩写
HtmlAnchor	hah
HtmlButton	hbtn
HtmlForm	hform
HtmlGenericControl	hgc
HtmlImage	himg
HtmlInputButton（按钮）	htxt
HtmlInputButton（重置）	hrcmd
HtmlInputButton（提交）	hccmd
HtmlInputCheckBox	hick
HtmlInputFile	hifile
HtmlInputHidden	hihidden
HtmlInputImage	hiimg

HtmlInputRadioButton	hirb
HtmlInputText（密码）	hpwd
HtmlInputText（文本）	hitxt
HtmlSelect	hslt
HtmlTable	htab
HtmlTableCell	htc
HtmlTableRow	htr
HtmlTextArea	htxta