保密级别：□绝密 □机密 □秘密 ■内部公开

**.NET编码规范**

天津三源电力智能科技有限公司

变更记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **修改点说明** | **变更日期** | **变更人** | **审批人** |
| V1.0 | 创建 | 2021.3.13 | 张婕 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

修改点说明的内容有如下几种：创建、修改（+修改说明）、删除（+删除说明）

一、  **环境设置**

首先去除VS开发环境中的一些选项如下：

粘贴时调整缩进

将类型的左大括号置于新行

将方法的左大括号置于新行

将匿名方法的左大括号置于新行

将控制块的左大括号置于新行

将“else”置于新行

将“catch”置于新行

将“finally”置于新行

复选框去掉.

二、  **命名规范**

1)        通用性

l   标识的总长度不要超过32个字符。

l     标识符的基本语法是以字母和\_开始，由字母数字及下划线组成的单词，第一个字符不能是数字。

l     只要合适，在变量名的末尾追加计算限定符（Avg、Sum、Min、Max、Index）。

l     在变量名中使用互补对，如 min/max、begin/end 和 open/close。

l     布尔变量名应该前加或包含 Is(is)。

l   尽量减少使用缩写，而是使用以一致方式创建的缩写。缩写应该只有一个意思；同样，每个缩写词也应该只有一个缩写。例如，如果用 min 作为 minimum 的缩写，那么在所有地方都应这样做；不要将 min 又用作 minute 的缩写。

l   在命名函数时包括返回值的说明，如 GetCurrentWindowName()。

l   避免对不同的元素重用名称，如名为 ProcessSales() 的例程和名为 iProcessSales 的变量。

l   在命名元素时避免同音异义词（如 write 和 right），以防在检查代码时发生混淆。

l   在命名元素时，避免使用普遍拼错的词。另外，应清楚区域拼写之间存在的差异，如 color/colour 和 check/cheque。

l   在内部范围中避免使用与外部范围中的名称相同的名称。若访问错误变量，则会产生错误结果。若变量与同一名称的关键字冲突，则必须在关键字前加适当的类型库以作标识。例如，若有一个名为 date 的变量，只能通过调用 System.Date 来使用内部 Date 函数。

l   接口名称以前缀“I”开始，后面接一个名词或名词词组（如 IComponent），或者接一个描述接口行为的形容词（如 IPersistable）。不要使用下划线，不要过多使用缩写，因为缩写会引起混淆。

l   事件处理程序的名称以一个描述事件类型的名词开始，后面接后缀“EventHandler”，如“MouseEventHandler”。 事件参数类的名称里要加“EventArgs”后缀。

l   如果某事件含有“之前”或“之后”的概念，请以现在时或过去时形式使用前缀，如“ControlAdd”或“ControlAdded”。

l   单个长字符串拆分成多行写。当一行被分为几行时，需要将串联运算符放在每一行的末尾。

l   SQL Server中不要给存储过程加sp 前缀/不要给用户定义的函数加 fn\_ 前缀/不要给扩展存储过程加 xp\_ 前缀。这些前缀是为标识系统保留的。将每个主要的SQL子句放在不同的行上，这样更容易阅读和编辑语句。

l   不要使用原义数字或原义字符串，如 For i = 1 To 7。而是使用命名常数，如 For i = 1 To NUM\_DAYS\_IN\_WEEK 以便于维护和理解。

2)        变量命名

变量名称命名规则：形容词+名词（或名词）

1. 属性（类属性/类属性对应的私有变量）

l  类属性与类属性对应的私有变量基本一样。

类属性对应的私有变量是在类属性名的前面加“\_”

如：private int \_PageSize;// 类属性对应的私有变量

public int PageSize { set { \_PageSize = value; } }//类属性

l  注意大小写要保持一致。每个单词的第一个字母必须大写。其它单词的第一个字母也大写。单词之间不加“\_”。

l  不要使用public来定义一个属性。

l  属性名和类名以名词开始，如 EmployeeName 和 CarAccessory。

1. 私有变量（短期性/长期性）

l  短期性（方法内私有变量/不是经常用的变量）

u  定义前加“\_”

u  如：string \_strSQL = null;

u  第一个单词的第一个字母必须小写，其它单词第一个字母大写。单词之间不加“\_”。

l  长期性（类私有变量/方法入口参数）

u  类私有变量：前加“\_”，和类属性对应的私有变量一样。每个单词的第一个字母必须大写。其它单词的第一个字母也大写。单词之间不加“\_”。

如：private int \_PageSizeTmp;

u  方法入口参数：第一个单词的第一个字母必须小写，其它单词的第一个字母必须大写。如果只有一个单词组成全小写。单词之间不加“\_”。

如：public static int SendCTTVOSMS(string mobile,string content)

public static string CallAccountHiVA(string restPhone,string userPhone)

1. 全局变量/静态变量/常量

l  定义要全部大写。如：public static int SMS\_TYPE = 2;

l  定义部分也可小写。

如：public static string VOSMS\_UserName = "88000002";

l  单词与单词之间加“\_”分隔。

3)        函数命名

函数命名规则：动词+名词（或动词），每个单词第一个字母必须大写。单词之间不加“\_”。

如：public static string GetOrderStatus(int sendMode,int statueID)

函数名和方法名以动词开始，如 InitNameArray() 和 CloseDialog()。

4)        控件命名

控件命名规则：类别+名称

类别对照表：

|  |  |
| --- | --- |
| 前缀 | 表示类型 |
| frm | 窗口 |
| btn | 按钮 |
| cbo | 下拉式列表框 |
| txt | 文本输入框 |
| lbl | 标签 |
| img | 图像 |
| pic | 图片 |
| div | DIV |
| grd | 网格 |
| scr | 滚动条 |
| lst | 列表框 |
| sds | SqlDataSource |
| ods | OleDbDataSource |

如按钮：btnSave

|  |  |
| --- | --- |
| 前缀 | 表示类型 |
| b/is | Bool |
| c | Char |
| sb | Sbyte |
| b | Byte |
| n/i | Int |
| ui | Uint |
| l | Long |
| ul | Ulong |
| f | Float |
| d | Double |
| s/str | String |

5)        表字段命名

l   在命名表时，用单数形式表示名称。例如，使用 Employee，而不是 Employees。

l   在命名表的列时，不要重复表的名称；例如，在名为 Employee 的表中避免使用名为 EmployeeLastName 的字段。

l   不要在列的名称中包含数据类型。如果后来有必要更改数据类型，这将减少工作量。

6)        Web文件目录结构命名

l   与过程名一样，文件和文件夹的名称也应该精确地说明它们的用途。

l   Web文件第一个单词的首字符要小写其它单词的首字符要大写。或全小写。文件存在后，在程序中要严格按照文件的大小写引入文件。目录名称必须全小写。

l   Web目录结构：

根---类库1

类库…N

解决方案启动文件

项目发布目录

    Web源代码--- inc( JS目录)

css(CSS目录)

后台目录

bin目录

app\_data数据库目录

master目录

其它子功能目录

app\_code类文件

images图片目录

三、  **注释规范**

1)        在文件的头部标明文件的作者，完成时间，它所完成的主要功能。

2)        程序有过改动后，要写上修改人、时间、简单原因说明列表。

如：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 谁创建的 日期 什么功能描述

\* 谁修改的 日期 什么功能描述

\* 谁修添加 日期 什么功能描述

\* 谁修删除 日期 什么功能描述

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

3)        函数等代码中的注释规范都按系统自动的注释格式

4)        修改代码时，总是使代码周围的注释保持最新。

5)        在每个例程的开始，提供标准的注释样本以指示例程的用途、假设和限制很有帮助。注释样本应该是解释它为什么存在和可以做什么的简短介绍。

6)        避免在代码行的末尾添加注释；行尾注释使代码更难阅读。不过在批注变量声明时，行尾注释是合适的；在这种情况下，将所有行尾注释在公共制表位处对齐。

7)        避免杂乱的注释，如一整行星号。

8)        在部署之前，移除所有临时或无关的注释，以避免在日后的维护工作中产生混乱。

9)        如果需要用注释来解释复杂的代码节，请检查此代码以确定是否应该重写它。尽一切可能不注释难以理解的代码，而应该重写它。尽管一般不应该为了使代码更简单以便于人们使用而牺牲性能，但必须保持性能和可维护性之间的平衡。

10)    在编写注释时使用完整的句子。注释应该阐明代码，而不应该增加多义性。

11)    在编写代码时就注释，因为以后很可能没有时间这样做。另外，如果有机会复查已编写的代码，在今天看来很明显的东西六周以后或许就不明显了。

12)    避免多余的或不适当的注释，如幽默的不主要的备注。

13)    使用注释来解释代码的意图。它们不应作为代码的联机翻译。

14)    注释代码中不十分明显的任何内容。

15)    为了防止问题反复出现，对错误修复和解决方法代码总是使用注释，尤其是在团队环境中。

16)    对由循环和逻辑分支组成的代码使用注释。这些是帮助源代码读者的主要方面。

17)    在整个应用程序中，使用具有一致的标点和结构的统一样式来构造注释。

18)     用空白将注释同注释分隔符分开。在没有颜色提示的情况下查看注释时，这样做会使注释很明显且容易被找到。

19)     为了防止在阅读代码时左右滚动源代码编辑器，每行代码或注释不得超过一个显示屏。

20)     可能多的注释变量表示的意思。

四、  **其它代码风格/习惯**

1)        JS和CSS文件必需是UTF-8编码的文件。

2)        单行的判断代码不需要加“{}”。

如：if (\_url.Trim().Equals(String.Empty)) return -2;

能简写的代码要简写。保证自己写出来的代码每一句都是有效代码。

3)        不要使用VS的自动排版代码功能。

4)        少用“==”运算。应该使用“.Equals”进行比较。

5)        运算符前后要空一格。如：\_TotalPage = \_TotalRecord / \_PageSize; 这样做是不会改变代码意图的，却可以使代码更加容易阅读。

6)        每一个操作结束后加一个空行。所有代码里不能有连续的二个或二个以上的空行。

7)        将大的复杂代码节分为较小的、易于理解的模块。

8)        近可能的使用TAB键来空位。不要使用4个空格来代替TAB键。

9)        在页面文件中不能定义static类型的来传递数据。