1.  命名规范

a) 类

【规则1-1】使用Pascal规则命名类名，即首字母要大写。

eg:

Class Test

{

    ...

}

【规则1-2】使用能够反映类功能的名词或名词短语命名类。

【规则1-3】不要使用“I”、“C”、“\_”等特定含义前缀。

【规则1-4】自定义异常类应以Exception结尾。

eg:

Class TestException

{

    ...

}

【规则1-5】文件名要能反映类的内容，最好是和类同名。

b) 类字段(类成员)

【规则2-1】用camel规则来命名类成员变量名称，即首单词（或单词缩写）小写。

【规则2-2】类字段变量名前可加“\_”前缀。

【规则2-3】坚决禁止在普通变量前加“m\_”（这是VC老命名规则）。

eg:

Class Test

{

    privatestring myName;

    privatestring \_myCoCo;

    ...

}

c）方法

   【规则3-1】方法名采用Pascal规则，第一个字符要大写。

【规则3-2】方法名应使用动词或动词短语。

【规则3-3】类中访问修饰符或功能相同的方法应该放在一起， 且公共或实现接口的方法在前。

eg:

Class Test

{

      ...

      publicvoid GetData(...)

      {

         ...

      }

      privatevoid GetName(...)

      {

         ...

      }

      privatestatic void GetCount(...)

      {

         ...

      }

}

   d）属性

      【规则4-1】使用名词定义属性，属性使用Pascal规则，首字符大写

      【规则4-2】属性和相应字段名称要关联， 可以使用“重构”菜单来生成属性。

eg:

Class Test

{

   private string myName;

   public string MyName

   {

      set

      {

         myName = Value;

      }

      get

      {

         return myName;

      }

   }

}

   e）参数

      【规则5-1】参数采用camel规则命名，且首字符小写。

      【规则5-2】使用描述性参数名称，参数名称应当具有最够的说明性。

      【规则5-3】不要给参数加匈牙利语类型表示法的前缀。

      【规则5-4】检查方法所有输入参数的有效性。

      eg:

      ClassTest

      {

         public void GetData(string name,string strFlag)

         {

            ...

         }

      }

   f）常量

      【规则6-1】只读常量使用Pascal命名规则，即首字母大写。

   【规则6-2】枚举名使用Pascal规则命名，枚举成员本质属于常量，命名规则同上。

      【规则6-3】枚举值从小到大顺序定义。

      【规则6-4】静态字段或属性采用Pascal规则，即首字符大写。

      eg:

      ClassTest

      {

         public const double Pi = 3.14159365753;

         private readonly double Pai = 3.14159365753;

      }

   g）接口

      【规则7-1】接口定义使用Pascal规则，且必须以大写“I”开头。

      【规则7-2】接口名称要有意义，中间不要有下划线“\_”等字符。

   【规则7-3】如果类实现了接口，名称尽量和接口相同， 只是省掉“I”字符。

   eg:注意这里的接口修饰符只能用public和internal

   interfaceITest

   {

      ...

   }

   h）事件

      【规则8-1】委托名称采用Pascal规则，即首字符大写。

   【规则8-2】定义事件的委托要使用EventHandler后缀，且包括sender和e两个参数。

   【规则8-3】事件用到的参数，名称要带EventArgs后缀。

   eg:

   ClassTest

   {

      private delegate void DoTask();

      private event DoTask DoTaskEventHandler;

      private event DoTask DoTaskEventHandler(Object sender,EventArgse);

      private event DoTaskDoTaskEventHandler2(string   strEventArgs)

      {

         ...

      }

      ...

   }

i) 命名空间

【规则9-1】命名空间名称采用Pascal规则，且首字符大写。

【规则9-2】命名空间名称尽量反映其内容所提供的整体功能。

eg:

namespace MyTest

{

      ClassTest

    {

       ...

    }

    ...

}

2.  注释规范

a）文件头部注释

【规则1-1】文件都包含文件头， 要说明文件名、作者、创建时间、变更记录。

      【规则1-2】推荐采用.NET形式书写头部注释。(待考查)

   b）类及其成员注释

      【规则2-1】对方法和类使用“///”三斜线注释。

   【规则2-2】代码行文注释采用“//”和“/\*\*/”进行，应该尽量说明问题。

3.行文规范

   a）缩写规范

      【规则1-1】标识符应当直观可望文知意，不提倡使用任何缩写。

   【规则1-2】字符串变量推荐是用“str”或“s”开头，采用string.Empty来初始化。

   【规则1-3】普通对象可以以“obj”开头。

   【规则1-4】缩写可自行定义，一般取单词的前/后字符组成，以含义直观为准则。

   【规则1-5】一般情况下不要让缩写破坏标识符的含义。

b）排版

   【规则2-1】每行语句至少占一行，如果语句过长（超过一屏），则该语句断为两行显示。

   【规则2-2】把相似的内容放在一起，比如字段、属性、方法、事件等，使用“#region--#endregion”命令分组。

   【规则2-3】多个程序元素进行对等操作时， 操作符之前、之后或者前后都要加空格。

   【规则2-4】每个方法的源程序行数原则上应该少于200行。(如果超过过多，则需要另写一方法)

   【规则2-5】语句嵌套层次不得超过3层。

   【规则2-6】避免相同的代码段在多个地方出现。(尽量避免代码重复，能复用则复用)

c）语句结构

   【规则3-1】如果使用了异常结构，一定要处理异常， 一般是要写日志文件。

   【规则3-2】分支语句不应该使用复杂长条件， 应该将长条件封装成方法。

   【规则3-3】switch语句，case后面必须接break。

   【规则3-4】禁止使用goto语句进行跳转。

   【规则3-5】行文中严禁出现“魔数”，特定含义的常数必须定义成枚举或常量。

   【规则3-6】不同类型的操作符混合使用时，使用括号给出优先级。

   【规则3-7】不允许使用复杂的操作符组合等。

   【规则3-8】循环、判断语句的程序块部分用花括号括起来， 即使只有一条语句。(return;待议)

   【规则3-9】在switch语句中总是要有default字句，建议使用断言。

   【规则3-10】每个类和方法完成单一的功能，不设计多用途面面俱到的类或方法。

   【规则3-11】严禁使用未经初始化的变量，变量通常使用构造方法来初始。

d）代码缩进

   【规则4-1】碰到大括号要换行。

   【规则4-2】不允许使用Java中的括号换行规范。

e）大小写

   【规则5-1】不能创建名称相同，但大小写区别的任何元素。

   【规则5-2】应当大写仅有两个字符的缩写。(如果只有两个字符，则都大写)

   【规则5-3】不要把易混淆的数字和字符放在一起。

   【规则5-4】使用英文命名标识符。

f）重名规范

   【规则6-1】不允许变量名、类名、属性名、 方法名等与系统标识符重名。

g）SQL编码规范

   【规则7-1】SQL语句全部大写。

   【规则7-2】对较为复杂的SQL语句加上注释，说明其功能。

   【规则7-3】连接符OR、IN、AND、以及=、<=、>=等前后加空格。

   【规则7-4】使用明确的列代替 SELECT \*。

h）软件架构

   【规则8-1】数据库中每一张表对应一个实体类/数据传输对象（DTO）。

   【规则8-2】实体类名称使用表名，也可带有Dto后缀。

   【规则8-3】三层架构应当合理使用，不应生搬硬套。

   【规则8-4】三层架构元素推荐使用后缀：

      数据传输对象         XxxxDto

      DAO工厂            XxxDAOFactory

      DAO接口            IXxxxDAO

      服务接口           IxxxxService

      DAO的数据库实现     XxxxDAOOracle/XxxxDAOInfomix

      业务逻辑           XxxxManager

附表

表1 各种类型命名规范总结

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **命名规则** | **注意事项** | **实例** |
| 类或结构 | Pascal | 首字符大写 | HttpContext |
| 接口 | Pascal | 加前缀I | IDataAdaper |
| 枚举名 | Pascal | 首字符大写 | CommandType |
| 枚举值 | Pascal | 首字符大写 | CommandType.Text |
| 事件 | Pascal | 首字符大写 | SelectedIndexChanged |
| 自定义异常 | Pascal | 加后缀Exception | ArgumentException |
| 类公共字段 | Pascal | 首字符大写 | MaxValue(或\_MaxValue) |
| 方法 | Pascal | 首字符大写 | ToString() |
| 命名空间 | Pascal | 首字符大写 | System.Xml |
| 属性 | Pascal | 首字符大写 | BackColor |
| 保护或私有字段 | Camel | 首字符小写 | myVariable |
| 参数 | Camel | 首字符小写 | cmdText |

表2 数据类型缩写规则

|  |  |  |
| --- | --- | --- |
| **数据类型** | **数据类型缩写** | **标准命名实例** |
| **Bool** | b/is | IsVisable |
| **Float** | F | FPrice |
| **Double** | D | DPrice |
| **Unit** | U | UAge |
| **Int** | I | INumber |
| **Char** | Ch | ChCode |
| **Byte** | Bt | BtImages |
| **String** | Str | StrName |
| **Struct** | St | StStudent |
| **Window** | Wnd | WndMain |
| **ArrayList** | Lst | LstStudents |
| **Array/strong>** | Arr | ArrStudents |
| **Hashtable** | Ht | Htstudents |

表 4 数据库对象缩写规范

|  |  |  |  |
| --- | --- | --- | --- |
| **数据库对象** | **名称** | **简写** | **实例** |
| **Connection** |  | Con | ConNorthwind |
| **Command** |  | Cmd | CmdReturnProducts |
| **Parameter** |  | Parm | ParmProductID |
| **DataAdapter** |  | Dap | DapProducts |
| **DataReader** |  | Dtr | DtrProducts |
| **DataSet** |  | Ds | DsNorthwind |
| **DataTable** |  | Dt | DtProduct |
| **DataRow** |  | Drow | DrowRow |
| **DataColumn** |  | Dcol | DcolProductID |
| **DataRelation** |  | Drl | DrlMasterDetail |
| **DataView** |  | Dvw | DvwFilteredProducts |