Code Style

1. 编码规范的作用：
2. 提高可读性。
3. 有助于工作交接，团队交流，提高工作效率。
4. 增强变量之间的关系，减少无关的变量引入，一次减少bug。
5. 提高程序员的个人能力并且有助于程序bug的修改。
6. 正视编码规范：

编码规范仅仅是一个全局性质的规范，它只不过是一种编程约定，不能解决更深层次的问题。当按照某种编码规范进行编程时，必然会有某些地方令人抓狂。在某些地方程序员的编码风格会优于这些规范。但是，这不重要。在某些地方，编码规范也有优于程序员的编程风格的时候。但是，这也不重要。只要这规范不是完全的不可理喻，在程序的可理解性上得到的好处会大大的补偿损失。

1. 编码规范的建议：
2. 将规范分为强制部分和推荐部分，将最基本的规范列放在强制部分，所有成员必须遵守；将好的但不重要的习惯列在推荐部分，开发人员可以根据自己习惯选择是否使用。
3. 编码规范不能太长。编码规范太长会消耗编程人员大量的精力。
4. 必须是约定俗成的 规范必须是行业中约定俗成的，不要有什么个性化发挥。

四、编码规范细则。

1、编码风格：

1） 严格采用阶梯层次组织程序代码：各层次缩进的分格采用VC的缺省风格，即每 层次缩进为4格。功能块、语句块的边界大括号一律独占一行，相匹配的大括号在同一列，对继行则要求再缩进4格；

2）花括号“{}”不允许省略，即使只有一段代码。

3）对变量的定义，尽量位于函数的开始位置；

4）使用括号“()”来强调运算符优先级。

5）各个大的功能块之间最好留一空行以及适当的注释；

6）不允许把多个短语句写在一行中，即一行只写一条语句；

7）不允许公开字段。

8）不允许省略访问修饰符。

9）函数名之后不要留空格，紧跟左括号‘（’，以与关键字区别；

10）‘（’向后紧跟，‘）’、‘，’、‘;’向前紧跟，紧跟处不留空格；

2、 命名规则：

2.1 常、变量命名

1）全局变量用g\_开头，即：变量名= g\_+变量类型+变量（或缩写） 静态变量用s\_开头，即：变量名= s\_+变量类型+变量（或缩写） 成员变量用m\_开头，即：变量名= m\_+变量类型+变量（或缩写）

2）常量和宏定义必须全部用大写字母，中间可根据意义的连续性用下划线连接， 每一个变量定义的右侧必须有一简单的注释；说明其作用；

3）用宏定义表达式时，要使用完备的括号。

4）局部变量——Camel风格

2.2类的命名

1）使用Pascal方式。在接口名称前加上前缀“I”。

2）考虑在派生类末尾使用基类的名字。

3）如果该类仅仅为了实现某个接口，那么请保持其与接口命名的统一

2.3 函数命名

1） 函数的命名应该尽量用英文表达出函数完成的功能，遵循动宾结构的命名法则： 成员函数可以只有动词，此时宾语默认为调用对象；

2） 引用外部函数必须注明函数来源（模块名及文件名）；

3） 函数参数规范：参数名称的命名参照变量命名规范；为了提高程序的运行效率， 减少参数占用的堆栈，传递大结构的参数，一律采用指针或引用方式传递；为了便于其他程序员识别某个指针参数是入口参数还是出口参数，同时便于编译器检查错误，应该在入口参数前加入const标志。

2.4数据库命名

1）表——“模块名\_表名”。

2）字段——bool类型用“Is”、“Can”、“Has”等表示；日期类型命名必须包含“Date”；时间类型必须包含“Time”。

3）存储过程——使用“proc\_”前缀。

4）视图——使用“view\_”前缀。触发器——使用“trig\_”前缀。

2.5 XML命名

1. 节点名称使用Pascal风格，属性名称使用Camel风格。
2. 注释风格：
3. 对接口和复杂代码块必须进行注释。
4. 修改代码时保持注释同步。
5. 未完成的功能使用TODO标记。
6. 修改他人代码时要先注释对方代码，并写明修改原因，不允许随便删除他人代码。
7. 发布前移除无用注释。
8. 异常处理
9. 原则上只允许显示抛出InvalidOperationException、ArgumentException、ArgumentNullException和ArgumentOutOfRangeException四种异常类型。
10. 在自定义异常时，必须使用VS提供的代码模板来创建自定义异常。