

山东三龙智能技术有限公司

C# 编码规范

（版本号：V0.1）

山东三龙智能技术有限公司上海研发中心

2013-4-1

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 修订目的描述 | 作者 |
| 2013-4-1 | 0.1 | 代码编写规范 | 李澍 |
|  |  |  |  |

目录

[1 概述 3](#_Toc352664833)

[1.1 必要性 3](#_Toc352664834)

[1.2 目的 3](#_Toc352664835)

[1.3 范围 3](#_Toc352664836)

[1.4 制定原则 3](#_Toc352664837)

[2 编码规范 3](#_Toc352664838)

[2.1 文件命名 3](#_Toc352664839)

[2.2 代码外观 4](#_Toc352664840)

[2.2.1 代码行的长度 4](#_Toc352664841)

[2.2.2 换行处理 4](#_Toc352664842)

[2.2.3 缩进 4](#_Toc352664843)

[2.2.4 空行 4](#_Toc352664844)

[2.2.5 空格 5](#_Toc352664845)

[2.2.6 括号 5](#_Toc352664846)

[2.2.7 花括号 5](#_Toc352664847)

[2.3 程序注释 6](#_Toc352664848)

[2.3.1 源文件注释 6](#_Toc352664849)

[2.3.2 类注释 7](#_Toc352664850)

[2.3.3 类属性注释 7](#_Toc352664851)

[2.3.4 类方法注释 8](#_Toc352664852)

[2.3.5 程序代码注释 8](#_Toc352664853)

[2.4 声明 8](#_Toc352664854)

[2.4.1 每行一个声明 8](#_Toc352664855)

[2.4.2 初始化 9](#_Toc352664856)

[2.4.3 位置安排 9](#_Toc352664857)

[2.4.4 类和接口声明 10](#_Toc352664858)

[2.4.5 字段声明 10](#_Toc352664859)

[2.5 命名规范 10](#_Toc352664860)

[2.5.1 命名空间命名 10](#_Toc352664861)

[2.5.2 类命名 11](#_Toc352664862)

[2.5.3 类成员变量命名 11](#_Toc352664863)

[2.5.4 接口命名 11](#_Toc352664864)

[2.5.5 属性命名 11](#_Toc352664865)

[2.5.6 枚举命名 11](#_Toc352664866)

[2.5.7 参数命名 12](#_Toc352664867)

[2.5.8 常量 12](#_Toc352664868)

[2.6 控件命名 12](#_Toc352664869)

[2.6.1 Windows空间命名规范 12](#_Toc352664870)

[2.6.2 ADO.NET控件命名规范 13](#_Toc352664871)

[2.7 程序版本组织 14](#_Toc352664872)

[2.7.1 主版本号 14](#_Toc352664873)

[2.7.2 次版本号 14](#_Toc352664874)

[2.7.3 Fix版本号 14](#_Toc352664875)

[2.7.4 样例 14](#_Toc352664876)

# 概述

## 必要性

1. 在软件生命周期中，70-80%的工作是在产品的维护上。
2. 在软件产品的整个生命周期中，他的维护不可能都由产品的最初开发人员来完成。
3. 良好的变成规范，可以增加程序的可读性，让软件开发人员更好更彻底的理解程序。
4. 当源代码作为产品的交付成果时，需要像其他产品一样，有着优美的视觉体验。

因此，要求软件开发人员要遵守约定的变成规范。

## 目的

为了统一公司软件项目中关于代码编写时的编写规范，保证代码的一致性，便于源代码的交流和维护，特制定此规范。

## 范围

本规范适用于参与软件项目开发的所有人员，作用于软件项目的开发阶段和后期维护阶段。

## 制定原则

1. 方便代码的交流与维护。
2. 不影响编码效率，不与大众习惯冲突。
3. 使代码更美观、更方便阅读。
4. 使代码的逻辑更清晰、更易于理解。

# 编码规范

## 文件命名

C# 源文件后缀为.cs，扩展名为小写。

## 代码外观

### 代码行的长度

代码行的长度控制在100个字符以内。

### 换行处理

当表达式超出或者即将超出规定的列宽，遵循以下规则进行换行

1. 在逗号后换行
2. 在操作符前换行
3. 规则1优先于规则2

### 缩进

采用4个空格作为代码的缩进，不要在代码中使用Tab字符。

### 空行

空行是为了将逻辑上相关联的代码分块，以便提高代码的可读性。

|  |  |
| --- | --- |
| 两个空行 | 1. 接口和类的定义之间 |
| 1. 枚举和类的定义之间 |
| 1. 类与类的定义之间 |
| 一个空行 | 1. 方法与方法、属性与属性之间 |
| 1. 方法中的变量声明与语句之间 |
| 1. 方法与方法之间 |
| 1. 方法中不同的逻辑块之间 |
| 1. 方法中的返回语句与其他语句之间 |
| 1. 属性与方法、属性与字段、方法与字段之间 |
| 1. 注释与他注释的语句不空行，单与其他的语句间空一行 |

### 空格

在以下情况中要使用到空格

1. 关键字和左括号“（”应该用空格隔开，如：

while (true)**;**

**注意**：在方法名和左括号“（”之间不需要使用空格，这样有助于 代码中的方法调用与关键字。

1. 多个参数用逗号“，”隔开，每个逗号后都应该加一个空格。
2. 除了“.”以外，所有的二元操作符都应该用空格与他们的操作数隔开，一元操作符、++ 以及 -- 与操作数之间不需要空格，如：

a **+=** c **+** d**;**

a **=** **(**a **+** b**)** **/** **(**c **\*** d**)**

**while** **(**d**++** **=** s**++)**

**{**

n**++;**

**}**

PrintSize**(**"size is" **+** size **+** "\n"**);**

语句中的表达式之间用空格隔开，如：

**for(**int i **=** 0**;** i **<** EscapeStr**.**Length**;** i**++)**

**{**

str **=** str**.**Replace**(**EscapeStr**[**i**],** EscapeToStr**[**i**]);**

**}**

### 括号

1. 左括号“（”不要紧靠关键字，中间用一个空格隔开。
2. 左括号“（”与方法名之间不要添加任何空格。

### 花括号

1. 左花括号“{”放于关键字或方法名的下一行并与之对齐。
2. 左花括号“{”要与相应的优化括号“}”对齐。
3. 通常情况下左花括号“{”单独成行，不与任何语句并列一行。

**if** **(**some\_condition**)**

**{**

**}**

**public** int Add**(**int x**,** int y**)**

**{**

**}**

1. if、while、do 语句后一定要使用 {}，即使{}中为空或者只有一条语句，如：

**if** **(**somevalue **==** 1**)**

**{**

somevalue **=** 2**;**

**}**

## 程序注释

### 源文件注释

单个文件开始以一下格式书写注释：

///<summary>

///Copyright (C) Shandong SARON Intelligent Technology Co., Ltd all rights reserved

///文件名称：Searh.aspx.cs

///功能概述：冷水表资料查询

/// 作者：李澍

///创建日期：2012-12-12

///</summary>

如果模块有修改，则每次修改要添加如下注释：

///<summary>

///Log 编号：<Log编号，从1开始递增>

///修改描述：<本次修改的简要描述>

///修改作者：<本次修改的作者>

///创建日期：2012-12-13

///</summary>

### 类注释

类注释参照如下格式：

///<summary>

///类的功能描述

///</summary>

### 类属性注释

///<summary>

///设备类型 <对此属性的描述性文字>

///</summary>

**public** string DeviceType

**{**

get

**{**

**return** **this.**DeviceType**;**

**}**

set

**{**

**this.**DeviceType **=** value**;**

**}**

**}**

### 类方法注释

/// <summary>

/// 检查数据包是否为注册包

/// </summary>

/// <param name="dataPacket">待检查数据包</param>

/// <param name="gprsId">返回GPRS标识，等于Invalid\_GPRSID为解析错误</param>

/// <returns>GPRS解析成功后返回true，否则false</returns>

**public** bool IsGPRSReg**(**byte**[]** dataPacket**,** **ref** ushort gprsId**)**

**{**

**}**

### 程序代码注释

1. 单行注释

// … (the source code) [代码注释原因][By:开发人员姓名][Date:YYYY-MM-DD]

1. 多行注释

/\*

[代码注释原因]

[By:开发人员姓名][Date:YYYY-MM-DD]

… (the source code)

\*/

## 声明

### 每行一个声明

因为鼓励写注释，所以我们建议最好每个声明占一行，例如：

int level**;** // 缩进级别

int size**;** // 表大小

要比：

int level, size**;**

更好。

### 初始化

在声明时初始化。只有在初始化值需要先做一些计算工作时，才不再声明之后立即初始化。

### 位置安排

变量建议置于块的开始处，不要总是在第一次使用他们的地方做声明。例如：

void MyMethod**()**

**{**

int int1 **=** 0**;** //在方法代码块开头声明

**if** **(**some\_condition**)**

**{**

int int2 **=** 0**;** //在if代码块开头声明

**...**

**}**

**}**

应该避免不同层级间的变量重名，如下是应该避免的：

int count**;**

**...**

void MyMethod**()**

**{**

**if** **(**some\_condition**)**

**{**

int count **=** 0**;** //应该避免

**...**

**}**

**...**

**}**

### 类和接口声明

1. 在方法名与其后的左括号间没有空格。
2. 左花括号“{”出现在声明的下行并与之对其，单独成行。
3. 方法间用一个空行隔开。

### 字段声明

尽量不要使用 public 或 protected 的实例字段。考虑为字段提供 get 和 set 属性访问器。

## 命名规范

### 命名空间命名

命名命名空间是的通用性规则是使用公司名称，后跟技术名称和可选的功能与设计，采用Pascal规则来命名，应该参照如下格式来命名：

CompanyName**.**TechnologyName**[.**Feature**][.**Design**]**

例如：

Saron**.**Meter**.**Basic**.**DataStruct

**注意**：命名空间命名时不要使用下划线、连字符或者任何其他非字幕的数字字符。

### 类命名

类名称应该是名词，可以由多个单词组成，使用Pascal规则，每个单词的首字母大写。类名要简单并且有描述性，尽量避免使用只取首字母的缩写词或缩略语（除非缩略语比长格式用的更广，比如说 URL或者HTML），在类命名中不要使用下划线字符（“\_”）。下面是几个正确的命名实例：

**public** class FileSteam**;**

**public** class ModbusProtocol**;**

**public** class BaseDataStruct**;**

### 类成员变量命名

类成员变量命名时，加前缀“m\_”（小写的m，代表member），后面采用Pascal规则，例如：

**int** m\_RecordCount**;**

**string** m\_DeviceName**;**

### 接口命名

接口名称以大写字母I开头，后跟名词活名词短语，使用Pascal规则，少用缩写和下划线（“\_”）。与类的命名规则的唯一区别就是多了前缀“I”。例如：

**public interface** IServiceProvider**;**

**public interface** IProtocal**;**

### 属性命名

应该是将后缀 Attribute 添加到自定义属性类。例如：

**public** class ObsoleteAttribute

**{**

**}**

### 枚举命名

枚举（Enum）应该遵循如下规则：

1. 对于Enum类型和值名称使用Pascal规则。
2. 尽量不用缩写。
3. 不要再Enum类型名称上使用Enum后缀。
4. 对大多数Enum类型使用单数名称，但是对作为位域的Enum类型使用复数名称。
5. 总是将FlagsAttribute 添加到位域 Enum 类型。

### 参数命名

1. 命名要有足够的描述性，便于理解其含义。
2. 对参数命名采用 Camel 规则。
3. 不要给参数名称添加匈牙利规则的前缀。

正确的命名例如：

**Type** GetType(**string** typeName)

### 常量

所有字母大写，单词之间用下划线（“\_”）连接，例如：

**public** const string COMPANY\_NAME **=** "SARON"

## 控件命名

### Windows空间命名规范

|  |  |  |
| --- | --- | --- |
| **类型** | **前缀** | **示例** |
| AdRotator | adrt | adrtTopAd |
| Button | btn | btnSubmit |
| Calendar | cal | calMettingDates |
| CheckBox | chk | chkBlue |
| CheckBoxList | chkl | chklFavColors |
| CompareValidator | valc | valcValidAge |
| CustomValidator | valx | valxDBCheck |
| DataGrid | dgrd | dgrdTitles |
| DataList | dlst | dlstTitles |
| DropDownList | drop | dropCountries |
| HyperLink | lnk | lnkDetails |
| Image | img | imgAuntBetty |
| ImageButton | ibtn | ibtnSubmit |
| Label | lbl | lblResults |
| LinkButton | lbtn | lbtnSubmit |
| ListBox | lst | lstCountries |
| Panel | pnl | pnlForm2 |
| PlaceHolder | plh | plhFormContents |
| RadioButton | rad | radFemale |
| RadioButtonList | radl | radlGender |
| RangeValidator | valg | valgAge |
| RegularExpression | vale | valeEmail\_Validator |
| Repeater | rpt | rptQueryResults |
| RequiredFieldValidator | valr | valrFirstName |
| Table | tbl | tblCountryCodes |
| TableCell | tblc | tblcGermany |
| TableRow | tblr | tblrCountry |
| TextBox | txt | txtFirstName |
| ValidationSummary | vals | valsFormErrors |
| XML | xmlc | xmlcTransformResults |

### ADO.NET控件命名规范

|  |  |  |
| --- | --- | --- |
| **类型** | **前缀** | **示例** |
| Connection | con | conNorthwind |
| Command | cmd | cmdReturnProducts |
| Parameter | parm | parmProductID |
| DataAdapter | dad | dadProducts |
| DataReader | dtr | dtrProducts |
| DataSet | dst | dstNorthWind |
| DataTable | dtbl | dtblProduct |
| DataRow | drow | drowRow98 |
| DataColumn | dcol | dcolProductID |
| DataRelation | drel | drelMasterDetail |
| DataView | dvw | dvwFilteredProducts |

## 程序版本组织

主程序的版本号 = 主版本号 + 次版本号 + Build（Fix版本号）

### 主版本号

表示程序的重大修改，例如，修改或新增了程序的重要功能。

### 次版本号

表示程序的一般修改，例如，修改或新增了程序的一般功能。

### Fix版本号

表示程序的Bug修改，以修改的日期缩写表示，例如：2013.4.1修改的话，Fix版本号为：20130401

### 样例

水资源管理系统版本号为：1.2 Build（20130401）