C#编码规范

1．命名规则与风格(naming conventions and style)

1．1命名规则

1．1．1 变量命名规范

|  |  |  |
| --- | --- | --- |
| 类型 | 前缀 | 示例 |
| Array | arr | arrShoppingList |
| Boolean | bln | blnIsPostBack |
| Byte | byt | bytPixelValue |
| Char | chr | chrDelimiter |
| DateTime | dtm | dtmStartDate |
| Decimal | dec | decAverageHeight |
| Double | dbl | dblSizeofUniverse |
| Integer | int | intRowCounter |
| Long | lng | lngBillGatesIncome |
| obxxxxject | obj | objReturnValue |
| Short | sht | shtAverage |
| Single | sng | sngMaximum |
| String | str | strFirstName |

1．1．2 控件命名规范

|  |  |  |
| --- | --- | --- |
| 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 |
| Hyperlixxxxnk | lnk | lnkDetails |
| Image | img | imgAuntBetty |
| ImageButton | ibtn | ibtnSubmit |
| Label | lbl | lblResults |
| lixxxxnkButton | lbtn | lbtnSubmit |
| ListBox | lst | lstCountries |
| Panel | pnl | pnlForm2 |
| PlaceHolder | plh | plhFormContents |
| RadioButton | rad | radFemale |
| RadioButtonList | radl | radlGender |
| RangeValidator | valg | valgAge |
| Regularexxxxxpression | vale | valeEmail\_Validator |
| Repeater | rpt | rptQueryResults |
| RequiredFieldValidator | valr | valrFirstName |
| Table | tbl | tblCountryCodes |
| TableCell | tblc | tblcGermany |
| TableRow | tblr | tblrCountry |
| TextBox | txt | txtFirstName |
| ValidationSummary | vals | valsFormErrors |
| xmxxxxl | xmxxxxlc | xmxxxxlcTransformResults |

1．1．3 ADO.NET控件命名规范

|  |  |  |
| --- | --- | --- |
| 类型 | 前缀 | 示例 |
| Connection | conn | connNorthwind |
| Command | cmd | cmdReturnProducts |
| Parameter | parm | parmProductID |
| DataAdapter | da | daProducts |
| DataReader | dtr | dtrProducts |
| DataSet | ds | dsNorthWind |
| DataTable | dt | dtProduct |
| DataRow | dr | drRow98 |
| DataColumn | dc | dcProductID |
| DataRelation | drel | drelMasterDetail |
| DataView | dvw | dvwFilteredProducts |

1．1．4 命名规则补充

Form frm  
 Class cls/C  
 Module mod  
 GroupBox grp  
 PictureBox pic  
 ComboBox cbo  
 TreeView tvw  
 ListView lvw  
 TabControl tab  
 DateTimePicker dtp  
 Timer tmr  
 Splitter spl  
 ProgressBar pbar  
 RichTextBox rtf  
 ImageList imgl  
 ToolBar tlb  
 MenuItem mnu  
 Date dat  
 Structure udt  
 Crystal Report rpt

1．1．5 事件处理子程序

The name of an event-handling subroutine will consist of the ID of the control that rasied the event followed by the type of event being handled. For example a subroutine named btnSubmit\_Click handles the Click event of a Button control named btnSubmit.

When a control that raises an event is not assigned an ID the type of the control is used instead of the ID. For example the subroutine named Button\_Click handles the Click event of a Button control without an ID.

1．2．编码风格

1 用pascal规则来命名方法和类型.

public class TextBox

{

public void DataBind()

{

}

}

2 用camel规则来命名局部变量和方法的参数.

string userName;

public AddUser(string userId byte[] password);

3 所有的成员变量前加前缀 \_

public class Databaxxxxse

{

private string \_connectionString;

}

4 接口的名称加前缀 I.

interface ICompare

{

int compare();

}

5 自定义的属性以Attribute结尾

public class AuthorAttribute : Attribute

{

}

6 自定义的异常以Exception结尾

public class AppException : Exception

{

}

7 方法的命名.一般将其命名为动宾短语.

ShowDialog()

CreateFile()

GetPath()

8 代码的缩进.要用Tab，而不要用space， 保持严格的缩进：

l 对缩进使用3个空格位

l 绝不使用不标准的缩进如12或4个空格位

9 局部变量的名称要有意义.不要用x，y，z等等(除用于For循环变量中可使用ijklmn).

string userName

10.建议局部变量在最接近使用它时再声明.

11 所有的成员变量声明在类的顶端，用一个换行把它和方法分开.

public class MyClass

{

int \_intNumber;

string \_strName;

public void SomeMethodl()

{}

public void SomeMethod2()

{}

}

12.用有意义的名字命名namespace，如：产品名、公司名.

13.避免使用命名空间的完整限定名称应使用using声明替代

14.避免using声明放置在命名空间内部

15.将所有的框架命名空间分组将用户或第三方命名空间放置在其下

using System;

using System.Collections;

using System.ComponentModel;

using System.Data;

using MyCompany;

using MyControls;

16.使用某个控件的值时，尽量命名局部变量.

17.文件名要能反应类的内容，最好是和类同名，一个文件中一个类或一组关连类.

18.大括号"{"要新起一行.

public class AuthorAttribute : Attribute

{

}

19.用C#预定义的类名，而不要用在System空间内别名.

obxxxxject 而不是 System.0bject

string 而不是 System.String

int 而不是 System.Int32

20.一行不要超过80个字符.

21.尽量不要手工更改机器生成的代码，若必须更改，一定要改成和机器生成的代码风格一样.

22.关键的语句(包括声明关键的变量)必须要写注释.

23.将注释缩拍在与代码缩进同等级的位置上

24.所有的注释应该通过拼写检查.拼错的注释将预示冗余的开发

25.文字常量和数字常量不要硬编码，应该用常量类或枚举代替.

26.不准使用goto系列语句.

27.不要声明public和protected的成员变量，应用property.

28.不要声明public的event，而应使用事件访问器.

public class Source

{

private EventHandler \_NumberChangeEvent;

public event EventHandler NumberChangeEvent

{

add

{

\_NumberChangeEvent += value;

}

remove

{

\_NumberChangeEvent -= value;

}

}

}

29.类型转换的使用规则.

Animal animal = new Dog();

Dog dog = animal as Dog;

if (dog != null)

{

}

30.生成和构建一个长的字符串时，一定要使用StringBuilder，而不用string.

31.始终使用"{ }"包含if下的语句，即使只有一条语句.

32.switch语句一定要有default来处理意外情况.

33.尽量少使用三目运算符 ? : ，而要使用if语句.

34.尽量不用使用this引用，除非是要调用类中的另一个Constructor.

public class Person

{

private string \_strName;

public Person(string name)

{

\_strName=name;

}

public Person() : this("Jim")

{

}

}

35.使用易描述的变量名称

l避免使用单一字符描述变量名称如i或t.应使用index或temp替代

l对public以及protected成员避免使用匈牙利命名法则

l尽量不使用缩写字符如使用num代替number

36. 对于泛型类型使用大写字母当处理.NET格式Type时使用Type后缀

//Correct:

public class lixxxxnkedList<K T>

{..}

//Avoid:

public class lixxxxnkedList<KeyTypeDataType>

{..}

37. 使用委托引用代替显式的委托实例(好像只有C#2.0才能这样用)

delegate void SomeDelegate();

public void SomeMethod()

{..

}

SomeDelegate someDelegate=SomeMethod;

38. 当使用partial类并将其分配给每个文件一部分时对每个文件命名使用后缀P并附带一个额外的数字

//In MyClassP1.cs

public partial class MyClass

{..}

// In MyClassP2.cs

public partial class MyClass

{..}

39.对于匿名方法参照有规律的代码规划其缩进应与匿名委托声明对齐

delegate void SomeDelegate (string someString);

//Correct:

public void InvokeMethod()

{

SomeDelegate someDelegate=delegate (string name)

{

MessageBox.Show(name);

};

someDelegate ("Juval");

}

//Avoid

public void InvokeMethod()

{

SomeDelegate someDelegate=delegate (string name)

{MessageBox.Show(name);};

someDelegate(”Juval");

}

40.对于匿名缺省参数的方法应该使用空括号表示.

delegate void SomeDelegate();

//Correct

SomeDelegate someDelegate1=delegate()

{

MessageBox.Show("Hello");

};

//Avoid

SomeDelegate someDelegate1=delegate

{

MessageBox.Show("Hello");

};

2．编码惯例(coding practices)

1. 避免在一个文件中放多个类。  
 2. 一个文件应该只对一个命名空间提供类型。避免在同一文件中有多个命名空间。  
 3. 避免文件长度超过500行(除了机器自动产生的代码)。  
 4. 避免方法定义超过25行。  
 5. 避免超过5个参数的方法。使用结构传递多个参数。  
 6. 每行应该不超过80个字符。  
 7. 不要手工编辑任何机器生成的代码。  
 a) 如果修改机器生成的代码，修改代码格式和风格以符合本编码标准。  
 b) 尽可能采用partial类以分解出需要维护的部分。

8. 避免对显而易见的内容作注释。

a) 代码应该是自解释的。 由可读性强的变量和方法组成的好的代码应该不需要注释。  
 9. 仅对操作的前提、内在算法等写文档。  
 10. 避免方法级的文档。  
 a) 对api文档采用大量的外部文档。

b) 方法级注释仅作为对其他开发人员的提示。  
 11. 决不要硬编码数值， 而总是声明一个常量。  
 12. 仅对本来就是常量的值使用const修饰符，例如一周的天数。  
 13. 避免对只读变量使用const修饰符。在此情况下，采用readonly修饰符。  
 public class myclass  
 {  
 public readonly int number;  
 public myclass(int somevalue)  
 {  
 number = somevalue;  
 }  
 public const int daysinweek = 7;  
 }  
 14. 对任何假设采用assert。  
 a) 平均地，每5行中就有一行是断言。  
 using system.diagnostics;  
 obxxxxject getobxxxxject()  
 {  
 obxxxxject obj = getobxxxxject();  
 debug.assert(obj != null);

}  
 15. 每行代码应该经过白盒测试。  
 16. 仅捕获已经显式处理了的异常。  
 17. 在抛出异常的catch语句中，总是抛出最初异常以保持最初错误的堆栈位置。  
 catch(exception exception)  
 {   
 messagebox.show(exception.message);  
 throw; //same as throw exception;  
 }  
 18. 避免将错误代码作为方法的返回值。  
 19. 避免定义自定义的异常类。  
 20. 定义自定义异常时：  
 a) 从ApplicationException继承  
 b) 提供自定义的序列化。  
 21. 避免在一个程序集中有多个main()方法。  
 22. 仅对最需要的类型标记为public，其他的标记为internal。  
 23. 避免采用friend程序集，因为这样增加了程序集间的耦合度。  
 24. 避免使用依赖于从特定位置运行的程序集的代码。  
 25. 尽量减少应用程序集(客户端exe程序集)的代码。采用类库而不要包含业务逻辑层代码。  
 26. 避免对枚举提供明确的值。  
 //correct   
 public enum color  
 {   
 redgreenblue  
 }  
 //avoid   
 public enum color  
 {   
 red = 1green = 2blue = 3  
 }  
 27. 避免对枚举指定类型。  
 public enum color : long  
 {   
 redgreenblue  
 }  
 28. if语句总是使用括号，即使它包含一句语句。  
 29. 避免使用?:条件算符。  
 avoid using the trinary conditional operator.   
 30. 避免在布尔条件语句中调用函数。赋值到局部变量并检查它们的值。  
 bool iseverythingok()  
 {...}  
 //避免：  
 if(iseverythingok())  
 {...}  
 //而应采用：  
 bool ok = iseverythingok();  
 if(ok)  
 {...}  
 31. 总是使用从0开始的数组。  
 32. 总是使用一个for循环显式地初始化一个引用类型的数组。  
 public class myclass  
 {}  
 myclass[] array = new myclass[100];  
 for(int index = 0; index < array.length; index++)  
 {  
 array[index] = new myclass();  
 }  
 33. 不要提供public或protected成员变量，而是使用属性。

34. 避免使用new继承修饰符，而是使用override。  
 35. 对非密封类总是将public和protected方法标记为virtual。  
 36. 除非涉及到互操作，永远不要用不安全的代码。  
 37. 避免显式类型转换。使用as算法防护性地转换类型。  
 dog dog = new germanshepherd();  
 germanshepherd shepherd = dog as germanshepherd;  
 if(shepherd != null)  
 {...}  
 38. 类成员有委托时：  
 a) 使用前将委托复制到局部变量，以避免并发冲突。  
 b) 调用前始终检查委托是否为空。  
 public class mysource  
 {  
 public event eventhandler myevent;  
 public void fireevent()  
 {  
 eventhandler temp = myevent;  
 if(temp != null)  
 {  
 temp(thiseventargs.empty);  
 }  
 }  
 }   
 40. 使用programming .net components中定义的eventshelper类安全地发布事件。   
 41. 总是使用接口。  
 42. 类和接口中方法和属性的比例至少是2：1。  
 43. 避免使用一个成员的接口。  
 44. 努力使每个接口拥有3-5个成员。  
 45. 每个接口不用超过20个成员。  
 a) 12可能是实际应用的极限了。  
 46. 避免将事件作为接口成员。  
 47. 避免使用抽象方法，而是使用接口代替。  
 48. 在类层次中暴露接口。  
 49. 优先使用明确的接口实现。  
 50. 永远不要假设一种类型支持某个接口。防护性地检查是否支持该接口。  
 sometype obj1;  
 Imyinterface obj2;  
   
 obj2 = obj1 as imyinterface;  
 if(obj2 != null)  
 {  
 obj2.method1();  
 }  
 else  
 {…}  
 51. 将呈现给用户的字符串永远不用硬编码，而是使用资源。  
 52. 发布时可能修改的字符串永远不用硬编码，例如连接字符串。  
 53. 构建一个长字符串时，使用stringbuilder，不要用string。  
 54. 避免提供带结构的方法。  
 a) 参数化的构造函数是鼓励使用的。  
 b) 可以重载算符。   
 55. 当提供静态成员变量时，总是提供一个静态构造函数。  
 56. 只要可以用前期绑定就不要用后期绑定。  
 57. 对应用程序进行日志和跟踪。  
 58. 除非在switch语句中跳转，永远不要用goto语句。  
 59. switch语句中总是使用default用于加断言。  
 int number = somemethod();  
 switch(number)  
 {  
 case 1:  
 trace.writeline("case 1:");  
 break;  
 case 2:  
 trace.writeline("case 2:");  
 break;  
 default:  
 debug.assert(false);  
 break;  
 }  
 60. 除非在构造函数中调用另一个构造函数，否则不用使用this。  
 public class myclass  
 {  
 public myclass(string message)  
 {}  
 public myclass() : this("hello")  
 {}  
 }  
 61. 除非为了解决调用基类构造函数时成员名的冲突，否则不要使用baxxxxse访问基类的成员。  
 public class dog  
 {  
 public dog(string name)  
 {}  
 virtual public void bark(int howlong)  
 {}  
 }  
 public class germanshepherd : dog  
 {  
 public germanshepherd(string name): baxxxxse(name)  
 {}  
 override public void bark(int howlong)   
 {  
 baxxxxse.bark(howlong);   
 }  
 }  
 63. 使用泛型的代码中避免与system.obxxxxject进行类型转换，而是使用限制或as算符。  
 class someclass  
 {}  
 //避免  
 class myclass   
 {   
 void somemethod(t t)   
 {  
 obxxxxject temp = t;   
 someclass obj = (someclass)temp;   
 }  
 }  
 //正确  
 class myclass where t : someclass  
 {   
 void somemethod(t t)   
 {  
 someclass obj = t;   
 }  
 }  
 64. 泛型接口不要定义限制。接口层的限制通常能用强类型代替。  
 public class customer  
 {...}  
 //避免  
 public interface ilist where t : customer   
 {...}  
 //正确  
 public interface icustomerlist : ilist   
 {...}  
 65. 不要在接口中定义与方法相关的限制。  
 66. 在数据结构中总是优先使用c#泛型。

3．框架特别指导(frxxxxamework specific guidelines)

1.总是以4级警告建立项目（图略）。

2． 在发布版中将警告作为错误（注意这不是VS.NET的缺省设置）（图略）。.  
 a) 虽然是可选的，本标准也推荐在调试版中将警告作为错误。

3. 永远不要抑制特定的编译警告（图略）。  
 4. 总是在应用程序的配置文件中显式地说明支持的运行时版本。  
 <?xmxxxxl version="1.0"?>  
 <configuration>  
 <startup>  
 <supportedRuntime version="v2.0.0.0"/>  
 <supportedRuntime version="v1.1.5000.0"/>  
 </startup>  
 </configuration>  
 5. 避免显式地自定义版本改向和绑定到CLR程序集。  
 6. 避免显式的预编译定义(#define)。使用项目设置定义条件编译常量。  
 7. 不要在AssemblyInfo.cs中放任何逻辑。  
 8. 除了在AssemblyInfo.cs，不要在任何文件中放程序集属性。  
 9. 在AssemblyInfo.cs中提供所有字段，例如公司名称、描述、版权等。  
 10. 所有程序集应该使用相对路径引用。  
 11. 不允许在程序集中循环引用。  
 12. 避免多模块的程序集。  
 13. 缺省总是以非检查的方式运行（为了性能考虑），但是对易于溢出或下溢的操作显式使用检查模式。  
 int CalcPower(int numberint power)  
 {  
 int result = 1;  
 for(int count = 1;count <= power;count++)  
 {  
 checked  
 {  
 result \*= number;  
 }  
 }  
 return result;  
 }  
 14. 避免使用Exception窗口（Debug|Exceptions）篡改异常处理。  
 15. 努力对同一逻辑应用程序中（通常是一个解决方案）的所有程序集和客户端使用统一的版本号。  
 16. Visual Studio.NET应用的配置文件命名为App.config，并将其包括在项目中。  
 17. 避免使用显式代码来排除方法(#if#endif)，而是使用条件方法。  
 public class MyClass  
 {  
 [Conditional("MySpecialCondition")]  
 public void MyMethod()  
 {}  
 }  
 18. 将VS.NET缺省的项目结构改为标准的布局，对项目文件夹和文件应用统一的结构。  
 19. 链接一个包含所有解决方案级信息的全局共享文件（图略）。  
 20. 制表符选用"插入空格"，使用3个空格代替制表符。  
 a) 在工具|选项|文本编辑器|C#|制表符中设置  
 21. 发布版中应该包含调试符号。  
 22. 总是对程序集签名，包括客户端应用程序。  
 23. 总是使用项目的SNK文件对互操作程序集签名（图略）。

4．项目设置和项目结构(Project Settings and Project Structure)  
 1. 总是使用类型安全的数据集或者数据表。避免使用原始的ADO.NET。  
 2. 访问数据库时总是使用事务。  
 a) 总是使用服务组件事务。  
 b) 不要使用ADO.NET事务。  
 3. 总是将事务隔离级别设置为序列的。  
 a) 使用其它隔离级别需要管理层决定。  
 4. 不要使用服务器浏览器将数据库连接拖到Windows表单、ASP.NET表单或Web服务中。这样做耦合了界面层和数据层。  
 5. 避免使用SQL Server验证。  
 a) 而是使用Windows验证。  
 6. 将访问SQL Server的组件以调用该组件客户端不同的身份运行。  
 7. 总是在高层用类型安全的类包装存储过程。仅在那个类中调用存储过程。  
 8. 避免在存储过程中放任何逻辑。  
 a) 如果存储过程中有IF，你可能犯了错误。