

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version V 1.0
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

C # 编码规范

Code Specification for C

(仅供内部使用 Only for inside of bpnet)

			文档编号 File No.	
文档名称 File Name	中文 Local	C# 编码规范	版本 Version	
	英文 English	Code Specification for C#	密级 Secret Level	
存放位置 Locate			作者 Author	

目录

一、目的	4
二、适用范围	4
三、命名空间	4
四、文件命名规范	5
五、控件命名规范	6
5.1 Win Form 内部控件命名规范	7
5.2 Web Form 内部控件命名规范	9
5.3 C# 数据组件命名规范	10
5.4 C# 菜单命名规范	10
六、变 / 常量命名规范	11
6.1 变量范围前缀	11
6.2 变量类型前缀	11
七、方法命名规范	13
八、代码注释规范	13
8.1 代码注释约定	13
8.2 模块头部注释规范	13
8.3 方法注释规范	15
8.4 代码行注释规范	18
8.5 变量注释规范	18
九、其它规范	19
9.1 编程风格	19
9.2 资源释放	21
9.3 错误处理：	22
9.4 其他	24

			文档编号 File No.	
文档名称 File Name	中文 Local	C# 编码规范	版本 Version	
	英文 English	Code Specification for C#	密级 Secret Level	
存放位置 Locate			作者 Author	

一、目的

1. 使用统一编码规范的主要原因，是使应用程序的结构和编码风格标准化，理解这段编码。
2. 好的编码约定可使源代码严谨、可读性强且意义清楚，与其它语言约定相一致，并且尽可能的直观。

[↑ 回目录](#)

二、适用范围

1. 本规范不适用于数据库开发规范，数据库开发相关规范请参考相关文档；
2. 本规范主要以 C# 为开发语言的规范，为开发部的原则性规范；
3. 由于本规范是为撰写程序而设计，所以适用于一切有关程序撰写的工作事项。对于具体的每个项目，可能需要对之进行裁剪和补存。补存的内容确认之后公用的内容要更新到此规范中；
4. 适用人员：所有开发人员及代码品管人员；
5. 适用产品：所有以 C# 编写的代码。

[↑ 回目录](#)

三、命名空间

命名空间应使用解决方案的名称，每个项目应设置一个二级命名空间，并以项目名命名。

如下图：

SDP 为解决方案 (Solution) 的名称，则 SDP 为顶级命名空间；

SDP.BF 为项目名称 (Assembly)，则 SDP.BF 项目下的命名空间为：SDP.BF.ClassName

			文档编号 File No.	
文档名称 File Name	中文 Local	C# 编码规范	版本 Version	
	英文 English	Code Specification for C#	密级 Secret Level	
存放位置 Locate			作者 Author	



[↑ 回目录](#)

四、文件命名规范

1. 文件命名原则是更容易区分不同的文件类型，在文件名前增加三字符的前缀，前缀字母一律为小写

例如：

一个窗体文件可以增加 frm 前缀，frmForm1.cs

所有的前缀列表请参考本小节末尾的表格！

2. 文件主体名必须用 名词或动名词，且主体名必须是单词首字大写的方式表示

例如：

销货单的窗体可以命名为 frmInvoice.cs，一张销货单批次作废的窗体可以命名为

frmCancelInvoice.cs

3. 文件名必须采用在不影响原意表达时尽量采用单词缩写的形式命名，

以达到文件名的简
Print Date:2017-03-11

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

洁明了的命名目的

例如：

应收帐款批次转凭证窗体的完整窗体名为 frmTransferAccountsReceivable.cs ，这时的窗体文件名太长，可以采用简写的方式，应收帐款专业简写为 AR ，我们可以采取这种公认的或专业的简写名词命名，最后可以命名为 frmTransAR

非标准缩写单词采用去掉元音字母或半元音字母的方式命名，重复字只取一个。例如：

Button : btn, 省图掉 u,o 和 t。

4. 文件名要和类名匹配

例如，对于类 HelloWorld, 相应的文件名应为 HelloWorld.cs (或, HelloWorld.vb)

5. 文件类型前缀一览表 (/ 表示暂无前缀)

Win Form		
扩展名 Extension	描述 File Name Description	前缀 Prefix
.CS	窗体文件	frm
.CS	类文件	/

Web Form		
扩展名 Extension	描述 File Name Description	前缀 Prefix
.aspx		/
.ascx	Web 用户自定义控件	wuc

 [回目录](#)

五、控件命名规范

1. C# 编码时，为了更直观地遍历整个窗体的控件，通常的作法是给每一种类型的控件的名称前增加统一的前缀。前缀字符采用三个小写字母的形式表示（一般常用的控件三个字母都可以满足，也有一些控件无法更好地用三个字母缩写，详见下面的表格。）

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

例如：

一个开始处理的按钮可以命名为：`btnBeginProc`

2. 控件主体名称采用名词或动名词的形式命名

例如：

1. 一个单号的标签：`lblDocNo`
2. 一个开始处理的按钮：`btnBeginProc`

3. 控件主体名采用单词首字大写的形式命名

例如：

一个产生应收帐款的按钮可以命名为：`btnMakeAR`

 [返回目录](#)

5.1 Win Form 内部控件命名规范

控件类型 Control Type	前缀 Prefix	例子 Example
Label	lbl	lblStartSync
LabelLink	lbk	
Button	btn	
TextBox	txt	
MainMenu	mnu	
CheckBox	chk	
RadioButton	rdo	
GroupBox	grp	
PictureBox	pic	
Panel	pan	
DataGrid	grd	
ListBox	lst	
CheckedListBox	chklst	
Combo	cbo	
ListView	lstv	

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

TreeView	trv	
TabControl	tab	
DateTimerPicker	dtp	
MonthCalendar	cld	
HScrollBar	hsb	
VScrollBar	vsb	
Timer	tmr	
Splitter	spl	
DomainUpDown	dup	
NumericUpDown	nup	
TrackBar	trk	
ProgressBar	pgr	
RichTextBox	rtxt	
ImageList	imglst	
HelpProvider	hlp	
ToolTip	tlp	
ContextMenu	mnu	
ToolBar	tlb	
StatusBar	sta	
NotifyIcon	nti	
OpenFileDialog	ofd	
SaveFileDialog	sfd	
FolderBrowserDialog	fbd	
FontDialog	fdg	
ColorDialog	cdg	
PrintDialog	pdg	
PrintPreviewDialog	ppd	
PrintPreviewControl	ppc	
ErrorProvider	erp	
PrintDocument	prd	
PageSetupDialog	psd	
CrystalReportViewer	crv	

 [返回目录](#)

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

5.2 Web Form 内部控件命名规范

控件类型 Control Type	前缀 Prefix	例子 Example
Label	lbl	lblTip
TextBox	txt	txtName
Button	btn	btnOK
LinkButton	lbtn	
ImageButton	ibtn	
HyperLink	hlk	
DropDownList	ddl	
ListBox	lst	
DataGrid	grd	
DataList	dlist	
Repeater	rep	
CheckBox	chk	
CheckBoxList	chklist	
RadioButtonList	rdolist	
RadioButton	rdo	
Image	img	
Panel	pan	
Placeholder	plh	
Calendar	clد	
AdRotator	adr	
Table	tbl	
RequireFieldValidator	rfv	
CompareValidator	cpv	
RangeValidator	rgv	
RegularExpressionValidator	rev	
CustomValidator	cstv	
ValidationSummary	vls	
Xml	xml	
Litteral	ltl	
CrystalReportViewer	crv	

 [返回目录](#)

		文档编号 File No.	
文档名称 File Name	中文 Local	C# 编码规范	版本 Version
	英文 English	Code Specification for C#	密级 Secret Level
存放位置 Locate			作者 Author

5.3 C# 数据组件命名规范

数据库对象 Data Object	前缀 Prefix	例子 Example
DataSet	ds	dsDept
DataTable	dt	dtDept
DataTableCollection	dtc	dtcDept
DataView	dv	dvDept
DataRow	dr	drDept
DataRowCollection	drc	drcDept
DataColumn	dc	dcField
DataColumnCollection	dcc	dccDept
DataRowView	drv	drvDept
OleDb Data Provider		
OleDbDataAdapter	oleadp	
OleDbConnection	oleconn	
OleDbCommand	olecmd	
Sql Server Data Provider		
SqlDataAdapter	sqladp	
SqlConnection	sqlconn	
SqlCommand	sqlcmd	
Odbc Data Provider		
OdbcDataAdapter	odbcadp	
OdbcConnection	odbcconn	
OdbcCommand	odbccmd	
Oracle Data Provider		
OracleDataAdapter	oraadp	
OracleConnection	oraconn	
OracleCommand	oracmd	

 [回目录](#)

5.4 C# 菜单命名规范

应用程序频繁使用许多菜单控件，对于这些控件具备一组唯一的命名约定很实用。除了最前面 "mnu" 标记以外，菜单控件的前缀应该被扩展：对每一级嵌套增加一个附加前缀，将

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

最终的菜单的标题放在名称字符串的最后。下表列出了一些例子。

菜单标题序列	菜单处理器名称
File Open	mnuFileOpen
File Send Email	mnuFileSendEmail
File Send Fax	mnuFileSendFax
Format Character	mnuFormatCharacter
Help Contents	mnuHelpContents

当使用这种命名约定时，一个特定的菜单组的所有成员一个接一个地列在 .net 的“属性”窗口中。而且，菜单控件的名字清楚地表示出它们所属的菜单项。

[↑ 回目录](#)

六、变 / 常量命名规范

6.1 变量范围前缀

1. 为了更好地区分变量的有效范围，例如 Class 级变量，在定义变量名时用 **小写字母前缀** 区分。
2. 此约定同样适用于 **常量的命名**。
3. **常量名必须大写**。

[↑ 回目录](#)

6.2 变量类型前缀

1. 定义变量时根据不同的变量类型增加特定的变量类型前缀，一般为小写的 **三个字母表示**
2. 变量主体名与控件主体名和窗体主体名的命名方式一致，采用 **单词首字大写的方式命名**

数据类型	CLR 类型	前缀	例子
Data type	CLR Type	Prefix	Example
bool	System.Boolean	bln	

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

byte	System.Byte	byt	
sbyte	System.SByte	sbyt	
char	System.Char	chr	
decimal	System.Decimal	dec	
double	System.Double	dbl	
float	System.Single	flt	
int	System.Int32	int	
uint	System.UInt32	uint	
long	System.Int64	lng	
ulong	System.UInt64	ulng	
short	System.Int16	sht	
ushort	System.UInt16	usht	
string	System.String	str	
datetime	System.DateTime	dtm	
object	System.Object	obj	
枚举		enu	
Structure		stru	

综上所述，变量命名格式为：< 特殊类型前导字 (都为小写字母) >< 类型前缀 >< 变量名字 >

特殊类型前导字：

前导字	数据类型
a	任何型态的数组（不区分为几维的数组）
m	Class 级变量前缀
I	所有接口声明及变量都应以大写 I 作为前缀
E	枚举声明增加 E 前缀

有效变量定义的例子：

`int[] aintCounters = new int[10];` （表示一个方法级的整型计数器数组）

`int[] maintCounters = new int[10];` （表示一个 class 级的整型计数器数组）

 [回目录](#)

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

七、方法命名规范

1. 方法名的主体应该使用大小写混合形式，并且应该足够长以描述它的作用。而且，方法名应该以一个动词起首，如 `InitNameArray` 或 `CloseDialog`。
2. 对于频繁使用的或长的项，推荐使用标准缩略语以使名称的长度合理化。一般来说，超过 32 个字符的变量名在 VGA 显示器上读起来就困难了。
3. 当使用缩略语时，要确保它们在整个应用程序中的一致性。在一个工程中，如果一会儿使用 `Cnt`，一会儿使用 `Count`，将导致不必要的混淆。

[↑ 返回目录](#)

八、代码注释规范

8.1 代码注释约定

1. 所有的方法和函数都应该以描述这段代码的功能的一段简明注释开始（方法是干什么）。这种描述不应该包括执行过程细节（它是如何做的），因为这常常是随时间而变的，而且这种描述会导致不必要的注释维护工作，甚至更糟——成为错误的注释。代码本身和必要的嵌入注释将描述实现方法。
2. 当参数的功能不明显且当过程希望参数在一个特定的范围内时，也应描述传递给过程的参数。被过程改变的函数返回值和全局变量，特别是通过引用参数的那些，也必须在每个过程的起始处描述它们。

[↑ 返回目录](#)

8.2 模块头部注释规范

以一个物理文件为单元的都需要，例如：`C#` 中包括 `cs` 檔

用于每个模块开头的说明，主要包括：**（粗体字为必需部分，斜体字为可选部分）**

1. 文件名称 (File Name) : 此文件的名称
2. 功能描述 (Description) : 此模块的功能描述与大概流程说明
3. 数据表 (Tables) : 所用到的数据表，视图，存储过程的说明，如关系比较复杂，则应说明哪些是可擦写的，哪些表为只读的。

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

4. 作者 (Author) :
5. 日期 (Create Date) :
6. 参考文档 (Reference) : 该档所对应的分析文档, 设计文档。
7. 引用 (Using) : 开发的系统中引用其它系统的 Dll、对象时, 要列出其对应的出处, 是否与系统有关 (不清楚的可以不写), 以方便制作安装档。
8. 修改记录 (Revision History) : 若档案的所有者改变, 则需要有修改人员的名字、修改日期及修改理由。

R1:
 修改作者 :
 修改日期 :
 修改理由 :

R2:
 修改作者 :
 修改日期 :
 修改理由 :

R3:
 ...

R10
 ...

R100

9. 分割符 : = = = = = (前后都要)

例图如下 :

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

```

// -----
//
// 文件名 (File Name):      ModuleBase.cs
//
// 功能描述 (Description):  所有Web User Control的基类。
//
// 数据表 (Tables):        nothing
//
// 作者 (Author):          Ben
//
// 日期 (Create Date):     2005.5.23
//
// 修改记录 (Revision History):
//
//   R1:
//     修改作者:            徐智雄
//     修改日期:            2005.10.24
//     修改理由:            增加全球化解决方案的公用方法。可以根据默认的语言或者特定的语言获取单个或多个资源信息；可以根据默认的语言或者特定的语言获取单个带替换参数的资源信息。
//
//   R2:
//     修改作者:            刘贞玲
//     修改日期:            2005.12.22
//     修改理由:            增加发布Web Message通知的公用方法。将要发送的通知内容按照模板的要素以资源ID的形式传入，此方法会根据194号系统参数的值发送相应的语言版本的通知。本方法根据传入参数来决定是否发送邮件通知还是发送联络单通知，或者是两者都发送。
//
//   R2:
//     修改作者:            Ben
//     修改日期:            2006.1.12
//     修改理由:            修改文件头部注释格式，增加修改记录部分的格式。
//
// -----
using System;
using System.Web.UI;

```

[↑ 回目录](#)

8.3 方法注释规范

注意事项：

1. 事件不需要头注解，但包含复杂处理时（如：循环 / 数据库操作 / 复杂逻辑等），应分割成单一处理函数，事件再调用函数。
2. 所有的方法必须在其定义前增加方法注释
3. 方法注释采用 `///` 形式自动产生 XML 标签格式的注释

```

/// <summary>
/// ...
/// </summary>
/// <param name= " " ></param>
/// <returns></returns>

```

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

标记	说明	备注
<c>	提供了一种将说明中的文本标记为代码的方法	
<code>	提供了一种将多行指示为代码的方法	
<example>	可以指定使用方法或其他库成员的示例。一般情况下，这将涉及到 <code> 标记的使用。	
<exception>	对可从当前编译环境中获取的异常的引用。	
<include>	得以引用描述源代码中类型和成员的另一文件中的注释。	
<list>	用于定义表或定义列表中的标题行。	
<para>	用于诸如 <summary> 、 <remarks> 或 <returns> 等标记内，使您得以将结构添加到文本中。	
<param>	应当用于方法声明的注释中，以描述方法的一个参数。	
<paramref>	提供了一种指示词为参数的方法。	
<permission>	得以将成员的访问记入文档。	
<remarks>	用于添加有关某个类型的信息，从而补充由 <summary> 所指定的信息。	
<returns>	应当用于方法声明的注释，以描述返回值。	
<see>	得以从文本内指定链接。	
<seealso>	对可以通过当前编译环境进行调用的成员或字段的引用。	
<summary>	应当用于描述类型或类型成员。	
<value>	得以描述属性。	

例图如下：

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

```

/// <summary>
/// 使用MD5计算哈希散列
/// </summary>
/// <param name="strSource">需要加密字符串</param>
/// <param name="strSaltCode">盐值</param>
/// <returns>加密后的字符串</returns>
public static string EncryptMD5(string strSource, string strSaltCode)
{
    //md5加密服务对象
    System.Security.Cryptography.MD5CryptoServiceProvider md5 =
        new System.Security.Cryptography.MD5CryptoServiceProvider();

    byte[] bytValue;
    byte[] bytHash;

```

4. 在公用类库中的公用方法需要在一般方法的注释后添加作者、日期及修改记录信息，统一采用 XML 标签的格式加注，标签如下：

```

<Author></Author>          作者
<CreateDate></CreateDate>    建立日期
<RevisionHistory>          修必记录
    <ModifyBy></ModifyBy>      修改作者
    <ModifyDate></ModifyDate>    修改日期
    <ModifyReason></ModifyReason> 修改理由

    <ModifyBy></ModifyBy>      修改作者
    <ModifyDate></ModifyDate>    修改日期
    <ModifyReason></ModifyReason> 修改理由

    <ModifyBy></ModifyBy>      修改作者
    <ModifyDate></ModifyDate>    修改日期
    <ModifyReason></ModifyReason> 修改理由
</RevisionHistory>
<LastModifyDate></LastModifyDate>    最后修改日期

```

5. 一个代码文件如果是由一人编写，则此代码文件中的方法无需作者信息，非代码文件作者在此文件中添加方法时必须添加作者、日期等注释，格式参考 8.4。
6. 修改任何方法，必须要添加修改记录的注释，格式参考 8.4。

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

8.4 代码行注释规范

1. 如果处理某一个功能需要很多行代码实现，并且有很多逻辑结构块，类似此种代码应该在代码开始前添加注释，说明此块代码的处理思路及注意事项等
2. 注释从新行增加，与代码开始处左对齐
3. 双斜线与注释之间以空格分开

例如：

```
public void Dispose()
{
    // 如果事务开启过，则释放事务对象
    if ( this.IDbTrans != null)
        this.IDbTrans.Dispose();

    // 如果连接已经打开，则关闭连接并释放资源
    if ( this.IDbConn.State == ConnectionState.Open )
    {
        this.IDbConn.Close();
        this.IDbConn.Dispose();
    }
}
```

 [回目录](#)

8.5 变量注释规范

1. 定义变量时需添加变量注释，用以说明变量的用途
2. class 级变量应以三条斜线的形式注释
3. 方法级的变量注释可以放在变量声明语句的后面，与前后行变量声明的注释左对齐，注释与代码间以 Tab 隔开。

例如：

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

```

3      /// <summary>
3      /// Database server name.
-     /// </summary>
-     private static string mstrServerName = "";
3      /// <summary>
3      /// The system database name of solomon.
-     /// </summary>
-     private static string mstrSysDBName = "";
3      /// <summary>
3      /// The login user name for the system database of solomon.
-     /// </summary>
-     private static string mstrSysUid = "";

protected bool SendNotification(string strSubjectResID,NameValueCollection objI
NameValueCollection objValues,string strLinkLabelResID,string strLink,
string strSendCOFrom,string strSendCOTo,string strSendMailTo,
bool blnIsSendMail,bool blnIsSendCO,MailPriority enuMailPriority)
{
    bool blnIsSuccess = false;           // 发送邮件是否成功

    string strRegard = "2835";           //2835(您好!)
    string strPrompt = "2836";           //2836(以上信息是由EPortal One自动发
    string strInfo = "2833";             //2833(有关信息如下)
    string strSubjectPrefix = "2853";    //2853(EPortal One Web Message)
}

```

[↑ 回目录](#)

九、其它规范

9.1 编程风格

1. 为了保持更好的阅读习惯， 请不要把多个变量声明写在一行中， 即一行只声明一个变量。

例如：

```
String strTest1,strTest2;
```

应写成：

```
String strTest1;
```

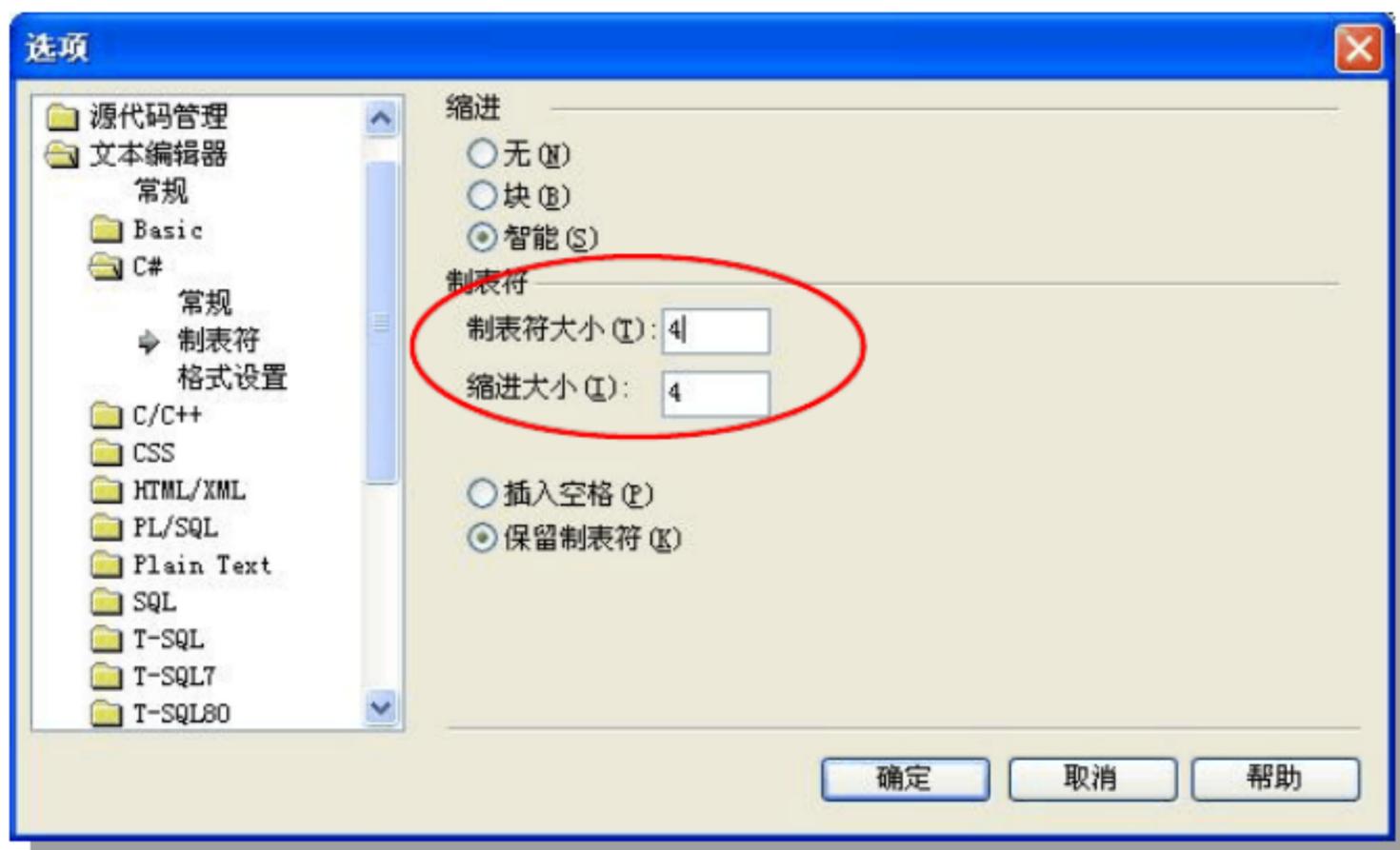
			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

String strTest2;

- 2 避免方法中有超过 5 个参数的情况。如果超过了，则应使用 struct 来传递多个参数。
- 3 为了更容易阅读，代码行请不要太长，最好的宽度是屏幕宽度（根据不同的显示分辨率其可见宽度也不同）。请不要超过您正在使用的屏幕宽度。（每行代码不要超过 80 个字符。）
- 4 除非在不完全的 switch 语句中否则不要使用 goto 语句。

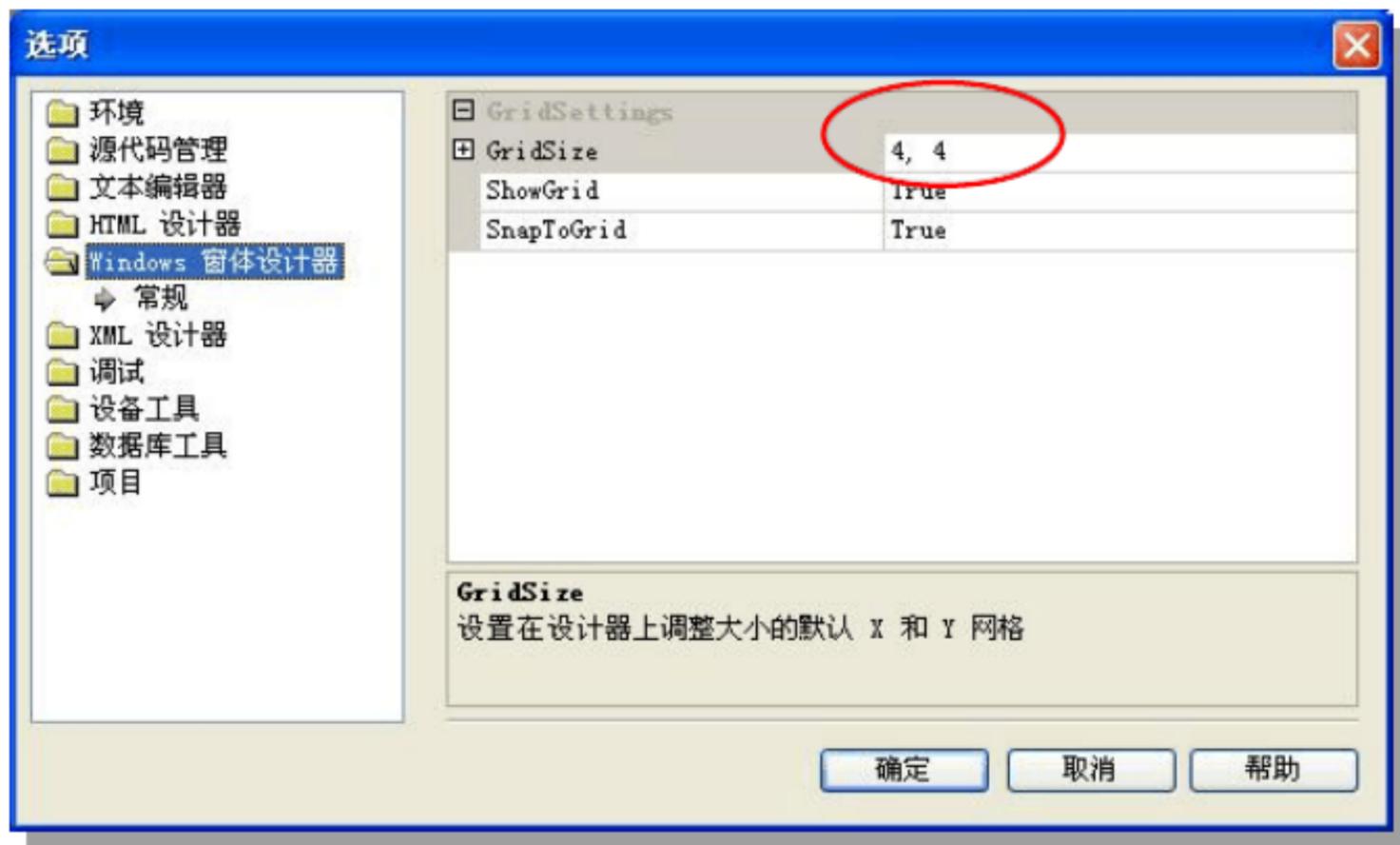
注：原则上不应使用 goto 语句，除非在能够大大减轻编码的复杂性，并不影响可读性的前提下才允许使用。

- 5 在 switch 语句中总是要有 default 子句来显示信息
- 6 代码缩进
 1. 一致的代码缩进风格，有利于代码的结构层次的表达，使代码更容易阅读和传阅
 2. 代码缩进请使用“TAB”键实现，不要使用空格，为保证在不同的机器上使代码缩进保持一致，特此规定 C# 的 TAB 键宽度为 4 个字符，设定界面如下 (C# Tools Option)：



7 界面排版

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	



- 8 方法参数多于 8 个时采用结构体或类方式传递
- 9 操作符 / 运算符左右空一个半角空格
- 10 所有块的 {} 号分别放置一行，并嵌套对齐，不要放在同一行上

[↑ 回目录](#)

9.2 资源释放

所有外部资源都必须显式释放。例如：数据库连接对象、IO 对象等。

			文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version	
	英文 English	Code Specification for C #	密级 Secret Level	
存放位置 Locate			作者 Author	

```

public void Dispose()
{
    // 如果事务开启过，则释放事务对象
    if ( this .IDbTrans != null )
        this .IDbTrans.Dispose();

    // 如果连接已经打开，则关闭连接并释放资源
    if ( this .IDbConn.State == ConnectionState.Open )
    {
        this .IDbConn.Close();
        this .IDbConn.Dispose();
    }
}

```

 [返回目录](#)

9.3 错误处理：

1. 不要“捕捉了异常却什么也不做”。如果隐藏了一个异常，你将永远不知道异常到底发生了没有。
2. 发生异常时，给出友好的消息给用户，但要精确记录错误的所有可能细节，包括发生的时间，和相关方法，类名等。
3. 只捕捉特定的异常，而不是一般的异常。

正确的做法：

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

```

void ReadFromFile ( string fileName )
{
    try
    {
        // read from file.
    }
    catch (FileIOException ex)
    {
        // log error.
        // re-throw exception depending on your case.
        throw ;
    }
}

```

不好：

```

void ReadFromFile ( string fileName )
{
    try
    {
        // read from file.
    }
    catch (Exception ex)
    {
        // Catching general exception is bad... we will never know
whether it
        // was a file error or some other error.

        // Here you are hiding an exception.
        // In this case no one will ever know that an exception happened.
        return "";
    }
}

```

		文档编号 File No.	
文档名称 File Name	中文 Local	C # 编码规范	版本 Version
	英文 English	Code Specification for C #	密级 Secret Level
存放位置 Locate			作者 Author

9.4 其他

1. 一个方法只完成一个任务。不要把多个任务组合到一个方法中，即使那些任务非常小。
2. 使用 C# 或 VB.NET 的特有类型，而不是 System 命名空间中定义的别名类型。
3. 别在程序中使用固定数值，用常量代替。
4. 避免使用很多成员变量。声明局部变量，并传递给方法。不要在方法间共享成员变量。
如果在几个方法间共享一个成员变量，那就很难知道是哪个方法在什么时候修改了它的值。
5. 别把成员变量声明为 public 或 protected 。都声明为 private 而使用 public/protected 的属性
6. 不在代码中使用具体的路径和驱动器名。使用相对路径，并使路径可编程。
7. 应用程序启动时作些“自检”并确保所需文件和附件在指定的位置。必要时检查数据库连接。出现任何问题给用户一个友好的提示。
8. 如果需要的配置文件找不到，应用程序需能自己创建使用默认值的一份。
9. 如果在配置文件中发现错误值，应用程序要抛出错误，给出提示消息告诉用户正确值。
10. DataColumn 取其列时要用字段名，不要用索引号。
例：正确 DataColumn[" Name"]
不好 DataColumn[0]
11. 在一个类中，字段定义全部统一放在 class 的头部，所有方法或属性的前面。
12. 在一个类中，所有的属性全部定义在一个属性块中：

```
#region Properties
```

```
#endregion
```