

# B4X 手册

B4A B4i B4J B4R

## B4X 语言

1	B4X 平台 .....	8
2	变量与对象 .....	9
2.1	变量类型 .....	9
2.2	变量名称 .....	12
2.3	声明变量 .....	12
2.3.1	简单变量 .....	12
2.3.2	数组变量 .....	13
2.3.3	常量变量 Const 关键字 .....	14
2.3.4	视图/节点 (对象) 数组 .....	15
2.3.5	类型变量 仅限 B4A, B4i 和 B4J .....	18
2.4	转换 .....	19
2.5	作为方法 .....	20
2.6	范围 .....	21
2.6.1	过程变量 .....	21
2.6.2	活动变量 仅限 B4A .....	22
2.6.3	局部变量 .....	22
2.7	提示 .....	22
3	程序流程 / 流程生命周期 .....	23
3.1	B4XPages 程序流程 .....	24
3.2	程序流程 B4A .....	25
3.2.1	程序启动 .....	26
3.2.2	过程全局变量 .....	27
3.2.3	活动变量 .....	27
3.2.4	启动服务 .....	28
3.2.5	程序流程 .....	29
3.2.6	Sub Process_Globals / Sub Globals .....	30
3.2.7	Sub Activity_Create (FirstTime As Boolean) .....	30
3.2.8	变量声明总结 .....	31
3.2.9	Sub Activity_Resume Sub Activity_Pause (UserClosed As Boolean) .....	32
3.2.10	Activity.Finish / ExitApplication .....	33
3.3	程序流程 B4i .....	34
3.4	程序流程 B4J .....	35
3.5	程序流程 B4R .....	36
3.6	程序流程比较 B4A / B4i / B4J .....	37
3.6.1	程序启动 B4A / B4i / B4J .....	37
3.6.2	旋转装置 B4A / B4i .....	37
4	B4X 语言 .....	38
4.1	表达式 .....	38
4.1.1	数学表达式 .....	38
4.1.2	关系表达式 .....	39
4.1.3	布尔表达式 .....	39
4.2	标准关键字 .....	40
	⊗ Abs (Number As Double) As Double .....	42
	⊗ ACos (Value As Double) As Double .....	42
	⊗ ACosD (Value As Double) As Double .....	42
	⊗ Array .....	42
	⊗ Asc (Char As Char) As Int .....	42
	⊗ ASin (Value As Double) As Double .....	42
	⊗ ASinD (Value As Double) As Double .....	42
	⊗ ATan (Value As Double) As Double .....	42
	⊗ ATan2 (Y As Double, X As Double) As Double .....	42

⊗ ATan2D (Y As Double, X As Double) As Double .....	42
⊗ ATanD (Value As Double) As Double .....	42
⊗ BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String .....	43
⊗ CallSub (Component As Object, Sub As String) As Object .....	43
⊗ CallSub2 (Component As Object, Sub As String, Argument As Object) As Object ..	43
⊗ CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object .....	43
⊗ CallSubDelayed (Component As Object, Sub As String) .....	43
⊗ CallSubDelayed2 (Component As Object, Sub As String, Argument As Object) .....	43
⊗ CallSubDelayed3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) .....	43
⊗ Catch .....	44
⊗ cE As Double .....	44
⊗ Ceil (Number As Double) As Double .....	44
⊗ Chr (UnicodeValue As Int) As Char .....	44
⊗ Continue .....	44
⊗ Cos (Radians As Double) As Double .....	44
⊗ CosD (Degrees As Double) As Double .....	44
⊗ cPI As Double .....	44
⊗ CreateMap .....	44
⊗ CRLF As String .....	45
⊗ Dim .....	45
⊗ Exit .....	45
⊗ Floor (Number As Double) As Double .....	45
⊗ For .....	45
⊗ If .....	46
⊗ IIf .....	46
⊗ Is .....	46
⊗ IsNumber (Text As String) As Boolean .....	46
⊗ LoadBitmap (Dir As String, FileName As String) As Bitmap .....	46
⊗ LoadBitmapResize (Dir As String, FileName As String, Width As Int, Height As Int, KeepAspectRatio As Boolean) As Bitmap .....	47
⊗ LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap .....	47
⊗ Log (Message As String) .....	47
⊗ Logarithm (Number As Double, Base As Double) As Double .....	47
⊗ LogColor (Message As String, Color As Int) .....	47
⊗ Max (Number1 As Double, Number2 As Double) As Double .....	47
⊗ Me As Object .....	47
⊗ Min (Number1 As Double, Number2 As Double) As Double .....	47
⊗ Not (Value As Boolean) As Boolean .....	48
⊗ Null As Object .....	48
⊗ NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String .....	48
⊗ NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String .....	48
⊗ Power (Base As Double, Exponent As Double) As Double .....	48

QUOTE As String.....	48
Regex As Regex.....	48
Return.....	48
Rnd (Min As Int, Max As Int) As Int.....	48
RndSeed (Seed As Long).....	48
Round (Number As Double) As Long 返回与给定数字最接近的长数字。.....	48
Round2 (Number As Double, DecimalPlaces As Int) As Double.....	49
Select.....	49
Sender As Object.....	49
Sin (Radians As Double) As Double.....	49
SinD (Degrees As Double) As Double.....	49
Sleep (Value As Double) As Double.....	49
Sqrt (Value As Double) As Double.....	49
SubExists (Object As Object, Sub As String) As Boolean.....	50
TAB As String.....	50
Tan (Radians As Double) As Double.....	50
TanD (Degrees As Double) As Double.....	50
True As Boolean.....	50
Try.....	50
Type.....	51
Until.....	51
While.....	51
4.3 条件语句.....	52
4.3.1 If - Then - Else.....	52
4.3.1.1 布尔求值顺序.....	53
4.3.2 IIf 内联如果.....	54
4.3.3 Select - Case.....	55
4.4 循环结构.....	57
4.4.1 For - Next.....	57
4.4.2 For - Each.....	58
4.4.3 Do - Loop.....	59
4.5 内联转换 As.....	61
4.6 子程序.....	62
4.6.1 声明.....	62
4.6.2 调用子程序.....	62
4.6.3 从另一个模块调用子程序.....	62
4.6.4 命名.....	63
4.6.5 参数.....	63
4.6.6 返回值.....	64
4.7 可恢复子程序.....	65
4.7.1 Sleep.....	65
4.7.2 Wait For.....	66
4.7.3 代码流.....	68
4.7.4 等待可恢复子程序任务完成.....	69
4.7.5 可恢复子程序返回值.....	70
4.7.6 B4A 仅限 KeyPress 和 Wait For MsgBox2Async.....	72
4.7.7 DoEvents 已弃用！.....	73
4.7.8 对话框.....	74
4.7.9 带有 Wait For 的 SQL.....	75
4.7.9.1 查询.....	75

4.7.9.2 B4J .....	76
4.7.10 注意事项和提示 .....	76
4.8 事件 .....	77
4.8.1 B4A .....	77
4.8.2 B4i .....	80
4.8.3 B4J .....	82
4.8.4 B4R .....	86
4.8.5 用户界面摘要 .....	87
4.9 库 .....	88
4.9.1 标准库 .....	89
4.9.2 附加库文件夹 .....	89
4.9.2.1 路径配置 B4A .....	90
4.9.2.2 路径配置 B4i .....	90
4.9.2.3 路径配置 B4J .....	91
4.9.2.4 路径配置 B4R .....	91
4.9.3 B4X 库 *.b4xlib .....	92
4.9.4 加载和更新库 .....	93
4.9.5 错误消息“您是否缺少库引用？” .....	93
4.9.6 在哪里可以找到库？ .....	94
4.9.6.1 在线库索引 .....	94
4.10 字符串操作 .....	95
4.10.1 B4A, B4i, B4J 字符串 .....	95
4.10.2 字符串连接 .....	96
4.10.3 B4A, B4i, B4J 字符串生成器 .....	97
4.10.3.1 StringBuilder 方法 .....	98
4.10.4 智能字符串文字 .....	99
4.10.4.1 字符串插值 .....	99
4.10.4.2 数字格式化程序 .....	99
4.10.4.3 其他格式化程序 .....	100
4.10.5 B4A, B4i 字符序列 CS 生成器 .....	101
4.10.5.1 文本 .....	101
4.10.5.2 使用 FontAwesome 或 MaterialIcons .....	103
4.10.5.3 图片 .....	103
4.10.5.4 可点击文本 .....	104
4.10.5.5 突出显示文本 .....	104
4.10.5.6 居中对齐文本 .....	105
4.10.5.7 CSBuilder 方法 .....	106
4.10.5.7.1 B4A / B4i .....	106
4.10.5.7.2 仅限 B4A .....	107
4.10.5.7.3 B4i only .....	108
4.10.6 B4J TextFlow 类 .....	109
4.10.7 B4R .....	110
4.11 数字格式 .....	113
4.11.1 B4A, B4i, B4J .....	113
4.11.2 B4X 数字格式化程序 .....	113
4.11.3 B4R .....	113
4.12 计时器 .....	114
4.13 文件 B4A, B4i, B4J .....	116
4.13.1 文件对象 .....	116
4.13.1.1 文件位置 .....	116
4.13.1.1.1 B4X .....	116
4.13.1.1.2 仅限 B4A .....	117

4.13.1.1.3	仅限 B4i .....	119
4.13.1.1.4	仅限 B4J .....	119
4.13.1.2	文件存在? B4A, B4i, B4J .....	120
4.13.1.3	常用方法 B4A, B4i, B4J .....	120
4.13.2	文件名 .....	122
4.13.3	子文件夹 .....	122
4.13.4	B4A, B4J TextWriter .....	123
4.13.5	B4A, B4J TextReader .....	124
4.13.6	文本编码 .....	125
4.14	列表 仅限 B4A, B4i 和 B4J .....	127
4.14.1	非动态列表 .....	129
4.15	地图 仅限 B4A, B4i 和 B4J .....	130
4.16	类模块 .....	132
4.16.1	入门 .....	132
4.16.1.1	添加类模块 .....	134
4.16.1.2	多态性 .....	135
4.16.1.3	自引用 .....	137
4.16.1.4	活动对象 仅限 B4A .....	137
4.16.2	标准类模块 .....	138
4.16.2.1	结构 .....	138
5	查找对象方法, 属性, 事件 .....	140
5.1	B4X 帮助查看器 .....	140
5.2	悬停在对象上 .....	141
5.3	定义事件例程 .....	142
6	代码片段 .....	143
6.1	在 IDE 中复制代码片段 .....	143
6.2	代码片段示例 .....	144
6.2.1	AdditionalLibraries\B4X\Snippets 文件夹中的简单代码片段 .....	144
6.2.2	b4xlib 库中的代码片段 .....	145
6.3	代码片段的位置 .....	147
6.3.1	在附加库文件夹下的特殊 Snippets 子文件夹中 .....	147
6.3.2	在 b4xlib 库中 .....	147
7	应避免的“代码异味”代码 .....	148
7.1	初始化一个对象, 然后将不同的对象赋给同一个变量 .....	148
7.2	已弃用的方法 - DoEvents, MsgBox .....	148
7.3	已弃用的方法 - Map.GetKeyAt / GetValueAt .....	148
7.4	File.DirDefaultExternal - 这始终是一个错误 .....	149
7.5	不使用参数化查询 .....	149
7.6	使用 Cursor 代替 ResultSet-Cursor .....	149
7.7	以编程方式构建完整布局 .....	149
7.8	重复代码 .....	150
7.9	不使用智能字符串的长字符串 .....	150
7.10	在不需要时使用全局变量 .....	150
7.11	当可用的时候不使用 Wait For .....	150
7.12	使用代码模块而不是类 .....	151
7.13	理解布尔值 .....	151
7.14	将“随机”字节转换为字符串 .....	151
7.15	手动生成或解析 XML/JSON .....	151
7.16	假设地图保持秩序 .....	152
8	Erel 建议避免的功能 .....	153
9	提示 .....	156
9.1	将数据与代码分离 .....	156

9.2	不要重复自己 (DRY 原则) .....	156
9.3	地图集合 .....	156
9.4	新技术和新特点。 .....	156
9.5	日志 .....	156
9.6	B4A 避免调用 DoEvents .....	157
9.7	字符串由字符而不是字节组成。 .....	157
9.8	B4A 使用服务, 尤其是入门服务 .....	157
9.9	UI 布局 .....	157
9.10	B4J 作为后端解决方案 .....	157
9.11	搜索 .....	158
9.12	Notepad++ .....	158
9.12.1	编码 .....	158

主要贡献者: Klaus Christl (klaus), Erel Uziel (Erel)

要搜索给定的单词或句子, 请使用“编辑”菜单中的“搜索”功能。

针对以下版本进行了更新:

B4A 版本 12.80

B4i 版本 8.50

B4J 版本 10.00

B4R 版本 4.00

#### [B4X 手册:](#)

B4X Getting Started

B4X 语言

B4X IDE Integrated Development Environment

B4X Visual Designer

B4X Help tools

B4XPages Cross-platform projects

B4X CustomViews

B4X Graphics

B4X XUI B4X User Interface

B4X SQLite Database

B4X JavaObject NativeObject

B4R 示例项目

您可以在此链接 [\[B4X\] 文档手册](#)中在线查阅这些手册。

请注意, 外部链接在在线显示中不起作用。

## 1 B4X 平台

B4X 是一套适用于不同平台的编程语言。

B4X 套件支持比任何其他工具更多的平台

ANDROID | IOS | WINDOWS | MAC | LINUX | ARDUINO | RASPBERRY PI | ESP8266 |  
和更多...

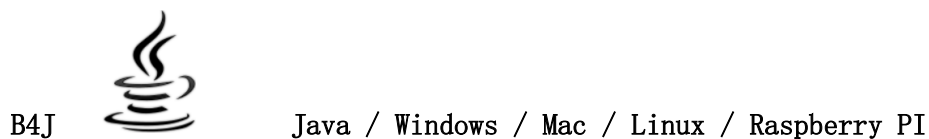


B4A 是一款 **100% 免费** 的安卓应用程序开发工具，它包括快速开发任何类型的安卓应用程序所需的所有功能。



B4i 是原生 iOS 应用程序的开发工具。

B4i 遵循与 B4A 相同的概念，允许您重用大部分代码并为安卓和 iOS 构建应用程序。



B4J 是一款 **100% 免费** 的桌面，服务器和物联网解决方案开发工具。

使用 B4J，您可以轻松创建桌面应用程序（UI），控制台程序（非 UI）和服务器解决方案。

- 编译后的应用程序可以在 Windows, Mac, Linux 和 ARM 板（如树莓派）。



B4R 是 **100% 免费** 的原生 Arduino 和 ESP8266 程序开发工具。B4R 遵循其他 B4X 工具的概念，提供简单而强大的开发工具。

B4R, B4A, B4J 和 B4i 共同构成物联网（IoT）的最佳开发解决方案。

### B4XPages

B4XPage 是 B4A, B4i 和 B4J 的内部库，允许轻松开发跨平台程序。

B4XPages 在 B4XPages 跨平台项目手册中有详细的解释。

即使您只想在一个平台上进行开发，使用 B4XPage 库也很有趣，它使程序流程更简单，尤其是对于 B4A。



2 变量与对象

**变量**是赋予某些已知或未知数量或信息的符号名称，目的是允许名称独立于它所代表的信息使用。计算机源代码中的变量名通常与数据存储位置相关联，因此也与它的内容相关联，这些变量名可能会在程序执行过程中发生变化（来源 Wikipedia）。

有两种类型的变量：原始类型和非原始类型。  
原语包括数字类型：Byte, Short, Int, Long, Float 和 Double。  
原语还包括：布尔值和字符。

2.1 变量类型

B4A, B4i, B4J

类型列表及其范围：

B4X	类型	最小值	最大值
Boolean	布尔值	False	True
Byte	整数 8 bits	$-2^7$ -128	$2^7 - 1$ 127
Short	整数 16 bits	$-2^{15}$ - 32768	$2^{15} - 1$ 32767
Int	整数 32 bits	$-2^{31}$ -2147483648	$2^{31} - 1$ 2147483647
Long	长整数 64 bits	$-2^{63}$ -9223372036854775808	$2^{63} - 1$ 9223372036854775807
Float	浮点数 32 bits	$-2^{-149}$ 1.4E-45	$(2 - 2^{-23}) * 2^{127}$ 3.4028235E38
Double	双精度数 64 bits	$-2^{-1074}$ 2.2250738585072014E- 308	$(2 - 2^{-52}) * 2^{1023}$ 1.7976931348623157E308
Char	字符		
String	字符数组		

B4R

类型列表及其范围：  
数字类型：

Byte	0 - 255	
Int (2 bytes)	-32,768 - 32,768	类似于其他 B4X 工具中的 Short 类型。
UInt (2 bytes)	0 - 65,535	B4R 专用。
Long (4 bytes)	-2,147,483,648 - 2,147,483,647	类似于其他 B4X 工具中的 Int 类型。
ULong (4 bytes)	0 - 4,294,967,295	B4R 专用。
Double (4 bytes)	4 字节浮点	类似于其他 B4X 工具中的 Float 类型。
Float 与 Double 相同。Short 与 Int 相同。		

以上在所有板上都是正确的，包括 Arduino Duo。

其他类型：  
**Boolean** True 或 False。 实际上，它被保存为一个值为 1 或 0 的字节。  
**String** 字符串由以空字节结尾的字节数组组成（值为 0 的字节）。  
**Object** 对象可以保存其他类型的值。

原始类型总是按值传递给其他子程序或分配给其他变量。例如：

```
Sub S1
    Private A As Int
    A = 12          变量 A = 12
    S2(A)          它按值传递给例程 S2
    Log(A) ' 打印 12  变量 A 仍然等于 12，即使 B 在例程 S2 中改变了。
End Sub

Sub S2(B As Int)  变量 B = 12
    B = 45          其值更改为 B = 45
End Sub
```

所有其他类型，包括原始类型数组和字符串，都归类为非原始类型。  
 当您将非原始类型传递给子程序或将其分配给不同的变量时，将传递引用的副本。  
 这意味着数据本身不会重复。  
 它与通过引用传递略有不同，因为您无法更改原始变量的引用。

所有类型都可以视为对象。  
 Lists 和 Maps 之类的 Collections 与 Objects 一起使用，因此可以存储任何值。  
 下面是一个常见错误的示例，其中开发人员试图将多个数组添加到列表中：

```
Private arr(3) As Int
Private List1 As List
List1.Initialize
For I = 1 To 5
    arr(0) = I * 2
    arr(1) = I * 2
    arr(2) = I * 2
    List1.Add(arr) '将整个数组添加为单个项目
Next
arr = List1.Get(0) '获取列表中的第一项
Log(arr(0))       '这里会打印什么???
```

您可能预计它打印 2。但是，它会打印 10。  
 我们创建了一个数组并将该数组的 5 个引用添加到列表中。  
 单个数组中的值是上次迭代中设置的值。  
 为了解决这个问题，我们需要在每次迭代时创建一个新数组。  
 这是通过每次迭代调用 Private 来完成的：

```
Private arr(3) As Int '在这种情况下，这个称呼是多余的。
Private List1 As List
List1.Initialize
For i = 1 To 5
    Private arr(3) As Int
    arr(0) = i * 2
    arr(1) = i * 2
    arr(2) = i * 2
    List1.Add(arr) '将整个数组添加为单个物品
Next
arr = List1.Get(0) '从列表中获取第一个物品
Log(arr(0)) '将打印 2
```

## 2.2 变量名称

除了保留字外，您可以为变量指定任何名称。

变量名必须以字母开头，并且必须由以下字符 A-Z，a-z，0-9 和下划线 “\_” 组成，不能有空格，不能有括号等。

变量名不区分大小写，这意味着 Index 和 index 指的是同一个变量。

但是给它们起有意义的名字是一种很好的做法。

例子：

```
Interest = Capital * Rate / 100  是有意义
n1 = n2 * n3 / 100              没有意义
```

对于视图 (B4A, B4i)，节点 (B4J)，在名称中添加一个定义其类型的三个字符的前缀很有用。

例子：

lblCapital	lbl > Label	Capital > 目的
edtInterest	edt > EditText	Interest > 目的
btnNext	btn > Button	Next > 目的

## 2.3 声明变量

### 2.3.1 简单变量

变量声明为 **Private** 或者 **Public** 关键词后跟变量名和 **As** 关键词然后是变量类型。详情请看[范围章节](#)。

存在着 **Dim** 关键词，这是为了兼容性而维护的。

例子：

<b>Private Capital As Double</b>	将三个变量声明为 Double， 双精度数。
<b>Private Interest As Double</b>	
<b>Private Rate As Double</b>	

<b>Private i As Int</b>	声明三个变量为 Int，整数。
<b>Private j As Int</b>	
<b>Private k As Int</b>	

<b>Private lblCapital As Label</b>	将三个变量声明为标签视图。
<b>Private lblInterest As Label</b>	
<b>Private lblRate As Label</b>	

<b>Private btnNext As Button</b>	将两个变量声明为按钮视图。
<b>Private btnPrev As Button</b>	

也可以用简短的方式声明相同的变量。

```
Private Capital, Interest, Rate As Double
Private i, j, k As Int
Private lblCapital, lblInterest, lblRate As Label
Private btnNext, btnPrev As Button
```

变量名用逗号分隔，后跟类型声明。

以下变量声明有效：

```
Private i = 0, j = 2, k = 5 As Int
```

```
Private txt = "test" As String, value = 1.05 As Double, flag = False As Boolean
```

如果我们想在代码中使用它们，就必须声明视图名称。

例如，如果我们要在代码中更改 Label 视图中的文本，例如

```
lblCapital.Text = "1200",
```

我们需要通过它的名字 lblCapital 来引用这个 Label view，这是通过 **Private** 声明完成的。

如果我们从未在代码中的任何地方引用此 Label 视图，则不需要声明。

对该视图使用事件例程也不需要声明。

要将值分配给变量，请写入其名称后跟等号再后跟值，例如：

```
Capital = 1200
```

```
LastName = "SMITH"
```

请注意，对于 Capital，我们只写了 1200，因为 Capital 是一个数字。

但是对于 LastName，我们写了 "SMITH"，因为 LastName 是一个字符串。

字符串必须始终写在双引号之间。

## 2.3.2 数组变量

数组是可以通过索引选择的数据或对象的集合。数组可以有多个维度。

声明包含 **Private** 或 **Public** 关键字，后跟变量名 LastName，方括号 (50) 之间的项目数，关键字 **As** 和变量类型 **String**。

有关详细信息，请参阅[范围章节](#)。存在 **Dim** 关键字，这是为了兼容性而维护的。

**注意：B4R 只支持一维数组！**

例子：

```
Public LastName(50) As String    一维字符串数组，物品总数 50。
```

```
Public Matrix(3, 3) As Double    二维数组 Doubles，物品总数 9。
```

```
Public Data(3, 5, 10) As Int     三维整数数组，物品总数 150。
```

数组中每个维度的第一个索引是 0。

```
LastName(0), Matrix(0,0), Data(0,0,0)
```

最后一个索引等于每个维度中的项目数减 1。

```
LastName(49), Matrix(2,2), Data(2,4,9)
```

```
Public LastName(10) As String
```

```
Public FirstName(10) As String
```

```
Public Address(10) As String
```

```
Public City(10) As String
```

或者

```
Public LastName(10), FirstName(10), Address(10), City(10) As String
```

此示例显示如何访问三维数组中的所有项目。

```
Public Data(3, 5, 10) As Int

For i = 0 To 2
  For j = 0 To 4
    For k = 0 To 9
      Data(i, j, k) = ...
    Next
  Next
Next
```

声明数组的一种更通用的方法是使用变量。

```
Public NbPers = 10 As Int
Public LastName(NbPers) As String
Public FirstName(NbPers) As String
Public Address(NbPers) As String
Public City(NbPers) As String
```

我们将变量声明为 `Public NbPers = 10 As Int` 并将其值设置为 10。然后我们用这个变量来声明数组，而不是像以前那样用数字 10 来声明。最大的优点是如果在某个时候我们需要更改项目的数量，我们只更改『一个』值。

对于 Data 数组，我们可以使用以下代码。

```
Public NbX = 2 As Int
Public NbY = 5 As Int
Public NbZ = 10 As Int
Public Data(NbX, NbY, NbZ) As Int
```

和访问例程。

```
For i = 0 To NbX - 1
  For j = 0 To NbY - 1
    For k = 0 To NbZ - 1
      Data(i, j, k) = ...
    Next
  Next
Next
```

使用 Array 关键字填充数组：

```
Public Name() As String
Name = Array As String("Miller", "Smith", "Johnson", "Jordan")
```

### 2.3.3 常量变量 Const 关键字

*Const* 变量是不能在代码中的任何地方更改的常量变量。为此，我们在 `Private` 或 `Public` 之后使用 `Const` 关键字，如下所示，

```
Private Const Size As Int = 10
Public Const ItemNumber As Int = 100
```

### 2.3.4 视图/节点（对象）数组

视图/节点或对象也可以在一个『数组』中。以下代码显示了一个示例：  
在 B4A 和 B4i 中用户界面对象称为*视图* (*views*) 而在 B4J 中称为*节点* (*nodes*)。

在下面的示例中，『按钮』(Button) 通过代码添加到父视图 / 节点。

#### B4A

##### Sub Globals

```
Private Buttons(6) As Button
End Sub
```

##### Sub Activity\_Create(FirstTime As Boolean)

```
Private i As Int

For i = 0 To 5
    Buttons(i).Initialize("Buttons")
    Activity.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
    Buttons(i).Tag = i + 1
    Buttons(i).Text = "Test " & (i + 1)
Next
End Sub
```

##### Sub Buttons\_Click

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```

#### B4i

##### Sub Process\_Globals

```
Private Buttons(6) As Button
End Sub
```

##### Private Sub Application\_Start (Nav As NavigationController)

```
Private i As Int
For i = 0 To 5
    Buttons(i).Initialize("Buttons")
    Page1.RootPanel.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
    Buttons(i).Tag = i + 1
    Buttons(i).Text = "Test " & (i + 1)
Next
End Sub
```

##### Sub Buttons\_Click

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```

**B4J****Sub Process\_Globals**

```
Private Buttons(6) As Button
End Sub
```

**Sub AppStart** (Form1 As Form, Args() As String)

```
Private i As Int
For i = 0 To 5
    Buttons(i).Initialize("Buttons")
    MainForm.RootPane.AddNode(Buttons(i), 10, 10 + i * 60, 150, 50)
    Buttons(i).Tag = i + 1
    Buttons(i).Text = "Test " & (i + 1)
Next
End Sub
```

**Sub Buttons\_MouseClicked** (EventData As MouseEvent)

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```

『按钮』也可以添加到布局文件中，在这种情况下，它们既不能被初始化，也不能被添加到父视图 / 节点，并且文本和标签属性也应该在设计器中设置。

在这种情况下，代码将如下所示：

**B4A****Sub Globals**

```
Private b1, b2, b3, b4, b5, b6, b7 As Button
Private Buttons() As Button
End Sub
```

**Sub Activity\_Create**(FirstTime As Boolean)

```
Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub
```

**Sub Buttons\_Click**

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```



**B4i****Sub Process\_Globals**

```
Private b1, b2, b3, b4, b5, b6, b7 As Button
Private Buttons(6) As Button
End Sub
```

**Private Sub Application\_Start** (Nav As NavigationController)

```
Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub
```

**Sub Buttons\_Click**

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```

**B4J****Sub Process\_Globals**

```
Private b1, b2, b3, b4, b5, b6, b7 As Button
Private Buttons(6) As Button
End Sub
```

**Sub AppStart** (Form1 As Form, Args() As String)

```
Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub
```

**Sub Buttons\_MouseClicked** (EventData As MouseEvent)

```
Private btn As Button
btn = Sender
Log("Button " & btn.Tag & " clicked")
End Sub
```

### 2.3.5 类型变量 仅限 B4A, B4i 和 B4J

类型不能是私有的。一旦声明它在任何地方都可用（类似于 Class 模块）。  
声明它们的最佳位置是在 Main 模块的 Process\_Globals 例程中。

让我们用一个人的数据重用这个例子。

我们可以使用 Type 关键字定义一个个人类型变量，而不是单独声明每个参数：

```
Public NbUsers = 10 As Int
Type Person(LastName As String, FirstName As String, Address As String, City As String)
Public User(NbUsers) As Person
Public CurrentUser As Person
```

新的个人类型是 `Person`，然后我们声明此个人类型的单个变量或数组。  
要访问特定项目，请使用以下代码。

```
CurrentUser.FirstName
CurrentUser.LastName
```

```
User(1).LastName
User(1).FirstName
```

变量名称，后跟一个点和所需的参数。

如果变量是一个数组，则名称后跟括号之间的所需索引。

可以将一个类型化变量分配给另一个相同类型的变量，如下所示。

```
CurrentUser = User(1)
```

## 2.4 转换

B4X 根据需要自动转换类型。它还自动将数字转换为字符串，反之亦然。

在许多情况下，您需要将 Object 显式转换为特定类型。

这可以通过将 Object 分配给所需类型的变量来完成。

例如，Sender 关键字引用一个对象，它是引发事件的对象。

以下代码更改按下按钮的颜色。

请注意，有多个按钮共享相同的事件子程序。

```
Sub Globals
    Private Btn1, Btn2, Btn3 As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
    Btn1.Initialize("Btn")
    Btn2.Initialize("Btn")
    Btn3.Initialize("Btn")
    Activity.AddView(Btn1, 10dip, 10dip, 200dip, 50dip)
    Activity.AddView(Btn2, 10dip, 70dip, 200dip, 50dip)
    Activity.AddView(Btn3, 10dip, 130dip, 200dip, 50dip)
End Sub

Sub Btn_Click
    Private btn As Button
    btn = Sender      ' 将对象转换成按钮
    btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

以上的代码也可以写得更优雅：

```
Sub Globals
End Sub

Sub Activity_Create(FirstTime As Boolean)
    Private i As Int
    For i = 0 To 9      ' 创建 10 个按钮
        Private Btn As Button
        Btn.Initialize("Btn")
        Activity.AddView(Btn, 10dip, 10dip + 60dip * i, 200dip, 50dip)
    Next
End Sub

Sub Btn_Click
    Private btn As Button
    btn = Sender      ' 将对象投射到按钮
    btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

## 2.5 作为方法

您可以使用 “As” 方法轻松地将一个对象转换为另一个对象。

当您想要将特定于平台的对象转换为跨平台对象时，这可能很有用，反之亦然。

例如，B4XView 确实存在 Rotation 属性，但对于 “标准” 标签就不存在。

```
Label1.As(B4XView).Rotation = 90
```

以上的行是短途，以下的三行做同样的事情，但很长：

```
Private xLabel1 As B4XView  
xLabel1 = Label1  
xLabel1.Rotation = 90
```

您还可以返回以设置特定于平台的属性：

```
xLabel1.As(Label).Padding(Array As Int(10dip, 0, 10dip, 0))
```

## 2.6 范围

### 2.6.1 过程变量

只要过程存在，这些变量就会存在。

您应该在 `Sub Process_Globals` 中声明这些变量。

这个 `sub` 在进程启动时被调用一次（这对所有模块都是如此，而不仅仅是主模块）。

这些变量是唯一的“公开”变量。这意味着它们也可以从其他模块访问。

但是，在 B4A 中，并非所有类型的对象都可以声明为流程变量。

例如，视图 / 节点不能声明为流程变量。

原因是我们不想持有对应该与活动一起销毁的对象的引用。

换句话说，一旦 `Activity` 被销毁，该 `Activity` 中包含的所有视图也将被销毁。

如果我们持有对视图的引用，垃圾收集器将无法释放资源，并且我们将发生内存泄漏。编译器强制执行此要求。

要访问其他模块中的进程全局变量，而不是声明它们的模块，它们的名称必须具有它们被声明为前缀的模块名称。

例子：

在名为 `MyModule` 的模块中定义的变量

```
Sub Process_Globals
    Public MyVar As String
End Sub
```

访问 `MyModule` 模块中的变量：

```
MyVar = "Text"
```

访问任何其他模块中的变量：

```
MyModule.MyVar = "Text"
```

变量可以声明为：

```
Dim MyVar As String
```

在这种情况下，变量是公开的，与 `Public` 相同。

像这样声明变量是一种很好的做法：

```
Public MyVar As String
```

这个变量是公开的。

可以像这样在 `Sub Process_Globals` 中声明私有变量：

```
Private MyVar As String
```

该变量对于声明它的 `activity` 或模块是私有的。

对于 `activity`，最好在 `Sub Globals` 中声明它们。

对于在 `Sub Class_Globals` 中的 `Class` 模块中声明的变量，与上述相同的规则是有效的。

```
Public MyVarPublic As String      ' 公共
Private MyVarPublic As String     ' 私有
Dim MyVar As String               ' 像公共一样公开 public like Public
```

不推荐在 `Sub Class_Globals` 中使用 `Dim` ！

### 2.6.2 活动变量 仅限 B4A

这些变量包含在活动中。

您应该在 Sub Globals 中声明这些变量。

这些变量是“私有的”，只能从当前活动模块访问。

所有对象类型都可以声明为活动变量。

每次创建活动时代，都会调用 Sub Globals（在 Activity\_Create 之前）。

只要活动存在，这些变量就存在。

### 2.6.3 局部变量

在子程序中声明的变量是该子程序的局部变量。

它们是“私有的”，只能从声明它们的子例程中访问。

所有对象类型都可以声明为局部变量。

在子程序的每次调用中，局部变量都被初始化为其默认值或您在代码中定义的任何其他值，并在退出子程序时被“销毁”。

## 2.7 提示

可以将视图 / 节点分配给变量，以便您可以轻松更改视图的通用属性。

例如，以下代码禁用作为面板 / 窗格的直接子级的所有视图：

```
For i = 0 To MyPanel.NumberOfViews - 1
    Private v As View
    v = MyPanel.GetView(i)
    v.Enabled = False
Next
```

如果我们只想禁用按钮：

```
For i = 0 To MyPanel.NumberOfViews - 1
    Private v As View
    v = MyPanel.GetView(i)
    If v Is Button Then ' 检查它是否是一个按钮
        v.Enabled = False
    End If
Next
```

注意：MyPanel 是 B4A 和 B4i 中的一个面板 (*Panel*)，但它是 B4J 中的一个窗格 (*Pane*)。

## 3 程序流程 / 流程生命周期

每个平台都有自己的程序流程。

为了制作跨平台项目，现在使用 B4XPages 更容易。

[B4XPages 跨平台项目手册](#)中详细解释了 B4XPages。

## 3.1 B4XPages 程序流程

对于具有 B4XPages 库的跨平台项目，所有三个平台的程序流程都是相同的。所有平台特定的代码都隐藏在 B4XPages 库中，对程序员是透明的。

B4XPages 跨平台项目手册中的 B4XPagesThreePages 项目显示了在页面之间导航时的程序流程。

例子：

项目启动，执行如下例程：

- |                       |        |
|-----------------------|--------|
| • MainPage Create     | 主页创建   |
| • MainPage Foreground | 主页前景   |
| • MainPage Appear     | 主页出现   |
| • MainPage Resize     | 主页调整大小 |

打开一个页面，示例中为 Page2：

- |                    |          |
|--------------------|----------|
| • Page2 Create     | Page2 创建 |
| • Page2 Foreground | Page2 前景 |
| • Page2 Appear     | Page2 出现 |

关闭页面，示例中为 Page2：

- |                   |          |
|-------------------|----------|
| • Page2 Disappear | Page2 消失 |
|-------------------|----------|



## 3.2 程序流程 B4A

让我们从简单的开始：

每个 B4A 程序都在自己的进程中运行。

一个进程有一个主线程，它也被称为 UI 线程，只要进程存在，它就会存在。一个进程也可以有更多的线程，这对后台任务很有用。

一个进程在用户启动您的应用程序时启动，假设它尚未在后台运行。

过程结束的决定性较小。它会在用户或系统关闭所有活动后的某个时间发生。

例如，如果您有一个活动并且用户按下后退键，则活动将关闭。稍后当手机内存不足（最终会发生）时，该过程将退出。

如果用户再次启动您的程序并且该进程没有被杀死，那么相同的进程将被重用。

B4A 应用由一项或多项活动组成。

**活动有点类似于 Windows 窗体。**

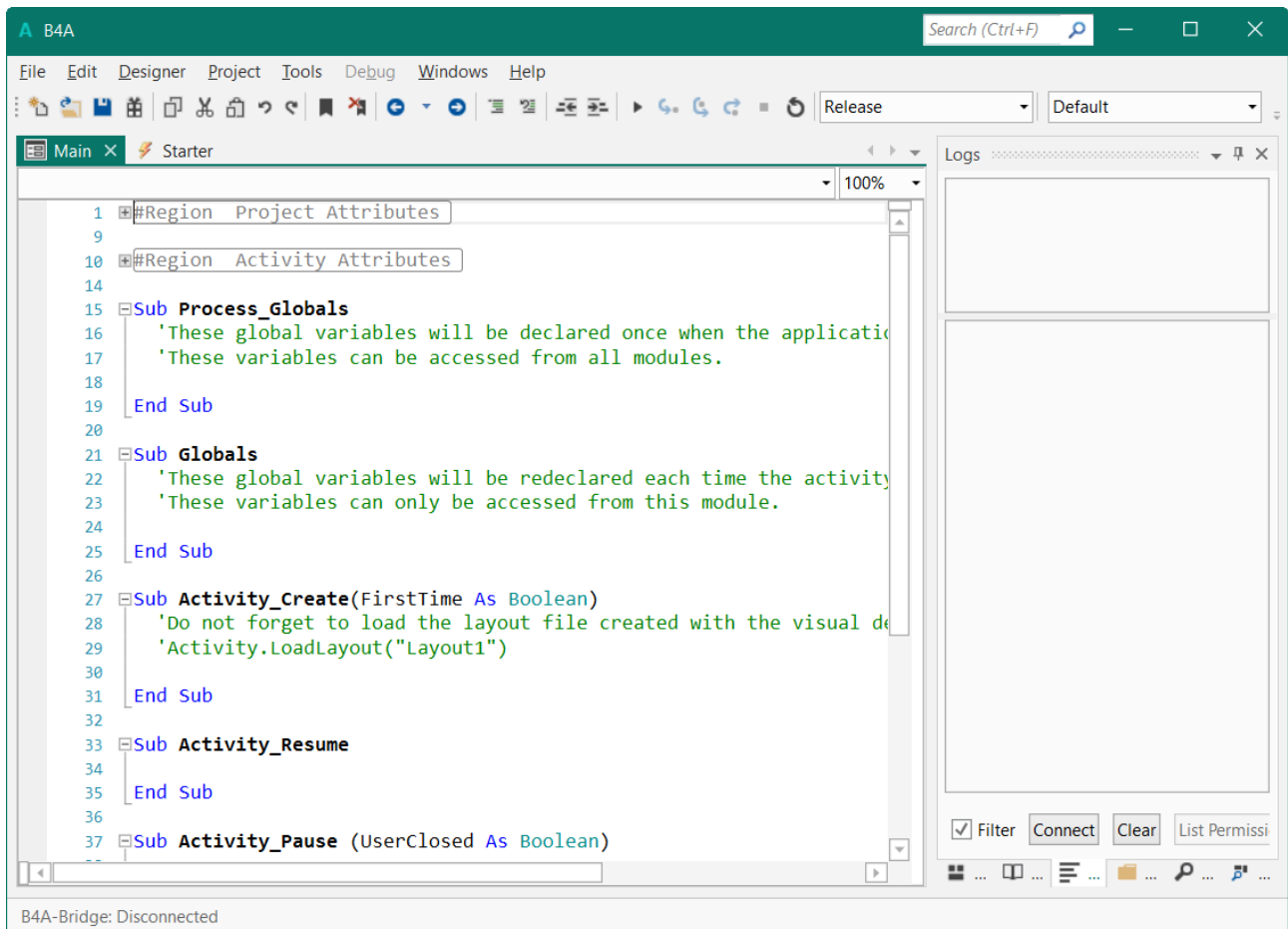
一个主要区别是，当一个活动不在前台时，它可以被杀死以保留内存。通常你会希望在活动丢失之前保存它的状态。在与进程关联的持久存储或内存中。


稍后将在需要时重新创建此活动。

当设备发生重大配置更改时，会发生另一个微妙的问题。最常见的是方向改变（用户旋转设备）。当发生这样的变化时，当前的活动将被销毁，然后重新创建。现在可以根据新配置创建活动（例如，我们现在知道新的屏幕尺寸）。

### 3.2.1 程序启动

当我们启动一个新程序时，我们得到以下模板：



在左上角，我们看到两个模块选项卡 ：

Main Activity

[Starter Service](#)

Starter Service 用于声明所有 ProcessGlobal 变量，并且可以从项目中的任何模块访问这些变量。

Main Activity 是起始 Activity，它不能被移除。

变量可以是全局的或局部的。局部变量是在 Process\_Globals 或 Globals 之外的 sub 中声明的变量。

局部变量是包含子或模块的局部变量。一旦 sub 结束，这些变量就不再存在。

可以从包含模块中的所有子访问全局变量。

有两种类型的全局变量。

流程变量（可从所有模块访问）和活动变量（可从单个模块访问）。

### 3.2.2 过程全局变量

只要过程存在，这些变量就会存在。

您应该在 Starter Service 的 Sub Process\_Globals 中将这些变量声明为 Public 似

**Sub Process\_Globals**

'这些全局变量将在应用程序启动时声明一次。

'可以从所有模块访问这些变量。

**Public** MyVariable = "Test" **As** String

该子程序在进程启动时被调用一次。

这些变量是唯一的“公共”变量。这意味着它们也可以从其他模块访问。

每个 Activity 模块中还有一个 Process\_Globals 例程。

如果您需要仅在 Activity 中有效的变量，它们仅在程序启动时初始化一次，您应该将它们放在 Activity 的 Process\_Globals 例程中（这适用于所有活动，而不仅仅是第一个活动）。

但是，并非所有类型的对象都可以声明为过程变量。

例如，所有视图都不能声明为过程变量。

原因是我们不想持有对应该与活动一起销毁的对象的引用。

换句话说，当 Activity 被销毁时，该 Activity 中包含的所有视图也会被销毁。如果我们不这样做，并且在 Activity 被销毁后保留对视图的引用，那么垃圾收集器将无法释放资源，并且会发生内存泄漏。

编译器强制执行此要求。

### 3.2.3 活动变量

这些变量归活动所有。

您应该在 Sub Globals 中声明这些变量。

这些变量是“私有的”，只能从当前活动模块访问。

所有对象类型都可以声明为活动变量。

每次创建活动时代，都会调用 Sub Globals（在 Activity\_Create 之前）。

只要活动存在，这些变量就存在。

### 3.2.4 启动服务

任何非小型 Android 应用程序的开发人员都需要应对的挑战之一是多个可能的入口点。

在几乎所有情况下的开发过程中，应用程序都将从 Main 活动开始。  
许多程序以类似于以下的代码开头：

```
Sub Activity_Create (FirstTime As Boolean)
    If FirstTime Then
        SQL.Initialize(...)
        SomeBitmap = LoadBitmap(...)
        '加载应用程序范围资源的附加代码
    End If
End Sub
```

在开发过程中，一切似乎都运行良好。然而，该应用程序“奇怪地”不时在最终用户设备上崩溃。这些崩溃的原因是操作系统可以从不同的活动或服务启动进程。例如，如果您使用 StartServiceAt 并且操作系统在后台终止该进程。  
现在 SQL 对象和其他资源将不会被初始化。

从 B4A v5.20 开始，有一个名为 Starter 服务的新功能，它提供了一个单一且一致的入口点。如果存在 Starter 服务，则该进程将始终从该服务启动。

将创建并启动 Starter 服务，然后才会启动应该启动的活动或服务。  
这意味着 Starter 服务是初始化所有应用程序范围资源的最佳位置。  
其他模块可以安全地访问这些资源。

Starter 服务应该是所有公共流程全局变量的默认位置。SQL 对象，从文件中读取的数据和多个活动使用的位图都应该在 Starter 服务的 Service\_Create 子程序中进行初始化。

#### 注意

- Starter 服务由其名称标识。您可以将名为 Starter 的新服务添加到现有项目中，它将成为程序入口点。  
这是通过选择项目 > 添加新模块 > 服务模块来完成的。
- 这是一项可选功能。您可以删除 Starter 服务。
- 如果您不希望服务继续运行，您可以在 Service\_Start 中调用 StopService(Me)。但是，这意味着该服务将无法处理事件（例如，您将无法使用异步 SQL 方法）。
- 启动服务应该从编译的库中排除。默认情况下，它的 #ExcludeFromLibrary 属性在服务属性区域中设置为 True。

### 3.2.5 程序流程

程序流程如下：

- **Main Process\_Globals**      主要模块的 Process\_Globals 例程  
在这里，我们为 Main 模块声明所有私有变量和对象。
- **Starter Service Process\_Globals** 如果服务存在，它就会运行。  
在这里，我们声明所有公共进程全局变量和对象，如 SQL，位图等。
- **其他 Activity Main Process\_Globals**      其他模块的 Process\_Globals 例程  
在这里，我们为给定模块声明所有私有变量和对象。
- **Starter Service Service\_Create** 如果服务存在，它就会运行。  
在这里，我们初始化所有公共进程全局变量和对象，如 SQL，位图等。
- **Starter Service Service\_Start**      如果服务存在，它就会运行。  
我们可以将这个例程留空。
- [Globals](#)  
在这里，我们为给定的 Activity 声明所有私有变量。
- [Sub Activity Create](#)  
这里我们加载布局并初始化代码添加的活动对象
- [Activity Resume](#)  
每次活动更改其状态时都会运行此例程。
- [Activity Pause](#)  
此例程在 Activity 暂停时运行，例如方向更改，启动另一个 Activity 等。

### 3.2.6 Sub Process\_Globals / Sub Globals

在任何 Activity 中，都应该使用 Process\_Globals 和 Globals 来声明变量。您还可以设置“简单”变量（数字，字符串和布尔值）的值。

你不应该在那里放任何其他代码。  
您应该将代码放在 Activity\_Create 中。

### 3.2.7 Sub Activity\_Create (FirstTime As Boolean)

创建活动调用此子程序。

活动已创建

- 当用户首次启动应用程序时
- 设备配置已更改（用户旋转设备）并且活动被破坏
- 当活动在后台并且操作系统决定销毁它以释放内存时。

这个子程序的主要目的是加载或创建布局（以及其他用途）。

FirstTime 参数告诉我们这是否是第一次创建此活动。第一次涉及到当前的进程。

您可以使用 FirstTime 运行与流程变量相关的各种初始化。

例如，如果您有一个包含需要读取的值列表的文件，则可以在 FirstTime 为 True 时读取它，并通过在 Sub Process\_Globals 中声明该列表将列表存储为流程变量

现在我们知道，只要进程存在，这个列表就可用，即使重新创建活动，也不需要重新加载它。

总而言之，你可以测试 FirstTime 是否为 True，然后初始化 Activity 的 Sub Process\_Globals 中声明的流程变量。

### 3.2.8 变量声明总结

我们应该声明哪个变量在哪里以及在哪里初始化我们的变量：

- 您想从多个模块访问的变量和无用户界面对象。  
像 SQL, 地图, 列表, 位图等。  
这些必须在 Starter Process\_Globals 中声明为 Public, 例如:

```
Sub Process_Globals
    Public SQL1 As SQL
    Public Origin = 0 As Int
    Public MyBitmap As Bitmap
End Sub
```

并在 Starter Service\_Create 中初始化, 如:

```
Sub Service_Create
    SQL1.Initialize(...)
    MyBitmap.Initialize(...)
End Sub
```

- 一个活动中的所有子程序都可以访问的变量, 应该只初始化一次。  
这些必须在 Activity Process\_Globals 中声明为 Private, 例如:

```
Sub Process_Globals
    Private MyList As List
    Private MyMap As Map
End Sub
```

并在 Activity\_Create 中初始化, 如:

```
Sub Activity_Create
    MyList.Initialize
    MyMap.Initialize
End Sub
```

- Class 或 Code module 中的变量  
这些大多被声明为 Private, 如果您希望它们可以从 Class 或 Code 模块外部访问, 您可以将它们声明为 Public。  
[B4X Booklet CustomViews Booklet](#) 中详细解释了类模块。
- 用户界面对象  
这些必须在 Activity 模块中声明, 它们在 Globals 中使用, 例如:

```
Sub Globals
    Private btnGoToAct2, btnChangeValues As Button
    Private lblCapital, lblInterest, lblRate As Label
End Sub
```

像 Int, Double String 和 Boolean 这样的简单变量可以直接在声明行中初始化, 即使在 Process\_Globals 例程中也是如此。

例子:

```
Public Origin = 0 as Int
```

不应在 Process\_Globals 例程中编写任何代码!

### 3.2.9 Sub Activity\_Resume Sub Activity\_Pause (UserClosed As Boolean)

Activity\_Resume 在 Activity\_Create 完成后或恢复暂停的活动后立即调用（活动移到后台，现在它返回到前台）。

请注意，当您打开不同的活动（通过调用 StartActivity）时，当前活动首先暂停，然后如果需要，将创建另一个活动并（始终）恢复。

每次活动从前台移动到后台时，都会调用 Activity\_Pause。

当 Activity 处于前台并且发生配置更改（这导致 Activity 暂停然后销毁）时，也会调用 Activity\_Pause。

Activity\_Pause 是保存重要信息的最后一个位置。

通常有两种机制可以让您保存活动状态。

仅与当前应用程序实例相关的信息可以存储在一个或多个流程变量中。

其他信息应存储在持久存储（文件或数据库）中。

例如，如果用户更改了某些设置，此时您应该将更改保存到持久存储中。否则更改可能会丢失。

每次活动从前台移动到后台时都会调用 Activity\_Pause。这可能是因为：

1. 开始了不同的活动。
2. 按下主页按钮。
3. 引发了配置更改事件（例如方向更改）。
4. 后退按钮被按下。

在场景 1 和 2 中，活动将被暂停并暂时保存在内存中，因为它预计稍后会被重用。

在场景 3 中，活动将被暂停，销毁，然后再次创建（并恢复）。

在场景 4 中，活动将被暂停并销毁。**按下返回按钮类似于关闭活动。**在这种情况下，您不需要保存任何特定于实例的信息（例如 pacman 在 PacMan 游戏中的位置）。

UserClosed 参数在这种情况下为真，在所有其他情况下为假。请注意，当您调用 Activity.Finish 时也是如此。该方法暂停并销毁当前活动，类似于后退按钮。

您可以使用 UserClosed 参数来决定要保存哪些数据以及是否将任何相关的流程变量重置为其初始状态（如果位置是流程变量，则将 pacman 位置移动到中心）。



### 3.2.10 Activity.Finish / ExitApplication

关于如何以及何时使用 Activity.Finish 和 ExitApplication 的一些解释。

可以在这里找到一篇关于 Android 功能的有趣文章：

[Android 方式的多任务处理](#)。

大多数应用程序不应该使用 ExitApplication 而是更喜欢 Activity.Finish 让操作系统决定何时终止进程。

仅当您确实需要完全终止该进程时才应使用它。

我们什么时候应该使用 Activity.Finish 什么时候不应该？

让我们考虑以下没有任何 Activity.Finish 的示例：

- **Main 活动**
  - StartActivity(SecondActivity)
- **SecondActivity 活动**
  - StartActivity(ThirdActivity)
- **ThirdActivity 活动**
  - 单击返回按钮
  - 操作系统返回上一个活动，SecondActivity
- **SecondActivity 活动**
  - 单击返回按钮
  - 操作系统返回之前的活动，Main
- **Main 活动**
  - 单击返回按钮
  - 操作系统退出程序

现在让我们考虑以下在每个 StartActivity 之前使用 Activity.Finish 的示例：

- **Main 活动**
  - Activity.Finish
  - StartActivity(SecondActivity)
- **SecondActivity 活动**
  - Activity.Finish
  - StartActivity(ThirdActivity)
- **ThirdActivity 活动**
  - 单击返回按钮
  - 操作系统退出程序

仅当我们不想使用 Back 按钮返回此活动时，我们才应该在开始另一个活动之前使用 Activity.Finish。

### 3.3 程序流程 B4i

B4i 中的程序流程比 B4A 程序流程简单得多。

当我们运行一个新项目时，我们会得到以下模板：

```
Sub Process_Globals
    'These global variables will be declared once when the application starts.
    'Public variables can be accessed from all modules.
    Public App As Application
    Public NavControl As NavigationController
    Private Page1 As Page

End Sub

Private Sub Application_Start (Nav As NavigationController)
    'SetDebugAutoFlushLogs(True) 'Uncomment if program crashes before all logs are
    printed.
    NavControl = Nav
    Page1.Initialize("Page1")
    Page1.Title = "Page 1"
    Page1.RootPanel.Color = Colors.White
    NavControl.ShowPage(Page1)
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)

End Sub

Private Sub Application_Background

End Sub
```

当您启动程序时，例程将按上述顺序执行。

请注意，Page1 的尺寸在 Application\_Start 中是未知的，它们仅在 Page1\_Resize 例程的 Width 和 Height 参数中已知。

如果您想调整视图，您必须在此处进行。

## 3.4 程序流程 B4J

B4J 中的程序流程比 B4A 程序流程简单得多，类似于 B4i。

当我们运行一个新项目时，我们会得到以下模板：

```
Sub Process_Globals
    Private fx As JFX
    Private MainForm As Form
End Sub

Sub AppStart (Form1 As Form, Args() As String)
    MainForm = Form1
    'MainForm.RootPane.LoadLayout("Layout1") '加载布局文件.
    MainForm.Show
End Sub

'返回 true 以允许默认异常处理程序处理未捕获的异常.
Sub Application_Error (Error As Exception, StackTrace As String) As Boolean
    Return True
End Sub
```

当您启动程序时，例程将按上述顺序执行。

如果要在用户调整表单大小时调整节点，则必须为此表单添加 Resize 例程，例如：

```
Private Sub MainForm_Resize (Width As Double, Height As Double)
    '你的代码
End Sub
```

如果您在设计器中使用锚点，则在大多数情况下不需要调整大小事件。

## 3.5 程序流程 B4R

B4R 中的程序流程是直截了当的。

当我们运行一个新项目时，我们会发现这个代码模板：

```
Sub Process_Globals
```

```
    ' 这些全局变量将在应用程序启动时声明一次。
```

```
    ' 可以从所有模块访问公共变量。
```

```
    Public Serial1 As Serial
```

```
End Sub
```

```
Private Sub AppStart
```

```
    Serial1.Initialize(115200)
```

```
    Log("AppStart")
```

```
End Sub
```

运行程序时，会执行 Process\_Globals，然后执行 AppStart。

Serial1.Initialize(115200)      初始化比特率。

Log("AppStart")                在日志中写入 “AppStart” 。

3.6 程序流程比较 B4A / B4i / B4J

3.6.1 程序启动 B4A / B4i / B4J

B4A	B4i	B4J
Main Process_Globals	Main Process_Globals	Main Process_Globals
Starter Process_Globals		
其他 modules Process_Globals	其他 modules Process_Globals	其他 modules Process_Globals
Starter Service_Create	Main Application_Start	Main AppStart
Starter Service_Start	Main Page1_Resize	Main MainForm_Resize
Main Globals		
Main Activity_Create FirstTime = True		
Main Activity_Resume		

3.6.2 旋转装置 B4A / B4i

B4A	B4i
Main Activity_Pause	
Main Globals	Main Page1_Resize
Main Activity_Create FirstTime = False	
Main Activity_Resume	

## 4B4X 语言

### 4.1 表达式

编程语言中的[表达式](#)是根据特定编程语言的特定优先级和关联规则解释的显式值，常量，变量，运算符和函数的组合，它计算然后产生（返回）另一个值。这个过程，就像数学表达式一样，被称为求值。该值可以是各种类型，例如数字，字符串和逻辑（来源 Wikipedia）。

例如， $2 + 3$  是算术和编程表达式，其计算结果为 5。变量是表达式，因为它是指向内存中值的指针，因此  $y + 6$  是表达式。关系表达式的一个示例是  $4 = 4$ ，其计算结果为 True（来源 Wikipedia）。

#### 4.1.1 数学表达式

运算符	例子	优先级	运算
+	$x + y$	3	添加
-	$x - y$	3	减法
*	$x * y$	2	乘法
/	$x / y$	2	分配
Mod	$x \text{ Mod } y$	2	模数
Power	$\text{Power}(x,y) \ x^y$	1	次方

优先级：在表达式中，级别 1 的操作在级别 2 的操作之前进行评估，而级别 2 的操作在级别 3 的操作之前进行评估。

例子：

$4 + 5 * 3 + 2 = 21$

>

$4 + 15 + 2$

$(4 + 5) * (3 + 2) = 45$

>

$9 * 5$

$(4 + 5)^2 * (3 + 2) = 405$

>

$9^2 * 5$

>

$81 * 5$

$\text{Power}(4 + 5, 2) * (3 + 2)$

$11 \text{ Mod } 4 = 3$

>

Mod 是 11 / 4 的余数

$23^3$

Power(23, 3)

>

23 的 3 次方

$-2^2 = -4$

$(-2)^2 = 4$

4.1.2 关系表达式

在关系表达式的计算机科学中，运算符测试两个实体之间的某种关系。 这些包括数值相等（例如， $5 = 5$ ）和不等式（例如， $4 \geq 3$ ）。

在 B4X 中，这些运算符返回 **True** 或 **False**，这取决于两个操作数之间的条件关系是否成立。

运算符	例子	用于测试
=	$x = y$	两个值的等价
<>	$x <> y$	两个值的否定等价
>	$x > y$	如果左边表达式的值大于右边的值
<	$x < y$	如果左侧表达式的值小于右侧表达式的值
>=	$x \geq y$	如果左表达式的值大于或等于右表达式的值
<=	$x \leq y$	如果左侧表达式的值小于或等于右侧表达式的值

4.1.3 布尔表达式

在计算机科学中，Boolean expression 布尔表达式是在计算时产生 Boolean 布尔值的表达式，即 **True** 或 **False** 之一。 布尔表达式可以由布尔常量 **True** 或 **False**，布尔类型变量，布尔值运算符和布尔值函数（来源 Wikipedia）的组合组成。

布尔运算符用于条件语句，例如 IF-Then 和 Select-Case。

运算符	评论
Or 或	布尔 或 $Z = X \text{ Or } Y$ $Z = \text{True}$ 如果 X 或 Y 等于 True 或 布两者都是 True
And 与	布尔 与 $Z = X \text{ And } Y$ $Z = \text{True}$ 如果 X 和 Y 都等于 True
Not ( ) 非	布尔 非 $X = \text{True}$ $Y = \text{Not}(X)$ $> Y = \text{False}$

		Or 或	And 与
X	Y	Z	Z
False	False	False	False
True	False	True	False
False	True	True	False
True	True	True	True

## 4.2 标准关键字

并非所有关键字都在 B4R 中可用。

-  [Abs](#) (Number As Double) As Double
-  [ACos](#) (Value As Double) As Double
-  [ACosD](#) (Value As Double) As Double
-  [Array](#)
-  [Asc](#) (Char As Char) As Int
-  [ASin](#) (Value As Double) As Double
-  [ASinD](#) (Value As Double) As Double
-  [ATan](#) (Value As Double) As Double
-  [ATan2](#) (Y As Double, X As Double) As Double
-  [ATan2D](#) (Y As Double, X As Double) As Double
-  [ATanD](#) (Value As Double) As Double
-  [BytesToString](#) (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String
-  [CallSub](#) (Component As Object, Sub As String) As Object
-  [CallSub2](#) (Component As Object, Sub As String, Argument As Object) As Object
-  [CallSub3](#) (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object
-  [Object](#)
-  [CallSubDelayed](#) (Component As Object, Sub As String)
-  [CallSubDelayed 2](#) (Component As Object, Sub As String, Argument As Object)
-  [CallSubDelayed 3](#) (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object)
-  [Object](#)
-  [Catch](#)
-  [cE](#) As Double
-  [Ceil](#) (Number As Double) As Double
-  [CharsToString](#) (Chars() As Char, StartOffset As Int, Length As Int) As String
-  [Chr](#) (UnicodeValue As Int) As Char
-  [Continue](#)
-  [Cos](#) (Radians As Double) As Double
-  [CosD](#) (Degrees As Double) As Double
-  [cPI](#) As Double
-  [CreateMap](#)
-  [CRLF](#) As String
-  [Dim](#)
-  [Exit](#)
-  [False](#) As Boolean
-  [Floor](#) (Number As Double) As Double
-  [For](#)
-  [GetType](#) (object As Object) As String
-  [If](#)
-  [Is](#)
-  [IsNumber](#) (Text As String) As Boolean
-  [LoadBitmap](#) (Dir As String, FileName As String) As Bitmap
-  [LoadBitmapResize](#) (Dir As String, FileName As String, Width As Int, Height As Int, KeepAspectRatio As Boolean) As Bitmap



-  [LoadBitmapSample](#) (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap
-  [Log](#) (Message As String)
-  [Logarithm](#) (Number As Double, Base As Double) As Double
-  [LogColor](#) (Message As String, Color As Int)
-  [Max](#) (Number1 As Double, Number2 As Double) As Double
-  [Me](#) As Object
-  [Min](#) (Number1 As Double, Number2 As Double) As Double
-  [Not](#) (Value As Boolean) As Boolean  [Null](#) As Object
-  [NumberFormat](#) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String
-  [NumberFormat2](#) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String
-  [Power](#) (Base As Double, Exponent As Double) As Double
-  [QUOTE](#) As String
-  [Regex](#) As Regex
-  [Return](#)
-  [Rnd](#) (Min As Int, Max As Int) As Int
-  [RndSeed](#) (Seed As Long)
-  [Round](#) (Number As Double) As Long
-  [Round2](#) (Number As Double, DecimalPlaces As Int) As Double
-  [Select](#)
-  [Sender](#) As Object
-  [Sin](#) (Radians As Double) As Double
-  [SinD](#) (Degrees As Double) As Double
-  [Sleep](#) (Milliseconds As Int)
-  [SmartStringFormatter](#) (Format As String, Value As Object) As String
-  [Sqrt](#) (Value As Double) As Double
-  [Sub](#)
-  [SubExists](#) (Object As Object, Sub As String) As Boolean
-  [TAB](#) As String
-  [Tan](#) (Radians As Double) As Double
-  [TanD](#) (Degrees As Double) As Double
-  [True](#) As Boolean
-  [Try](#)
-  [Type](#)
-  [Until](#)
-  [While](#)

### **Abs** (Number As Double) As Double

返回绝对值。

### **ACos** (Value As Double) As Double

计算三角反余弦函数。 返回用弧度测量的角度。

### **ACosD** (Value As Double) As Double

计算三角反余弦函数。 返回用度数测量的角度。

### **Array**

创建指定类型的一维数组。

语法是: Array [As type] (值列表)。

如果省略类型, 则将创建一个对象数组。

例子:

```
Dim Days() As String
Days = Array As String("Sunday", "Monday", ...)
```

### **Asc** (Char As Char) As Int

返回给定字符或字符串中第一个字符的 unicode 代码点。

### **ASin** (Value As Double) As Double

计算三角反正弦函数。 返回用弧度测量的角度。

### **ASinD** (Value As Double) As Double

计算三角反正弦函数。 返回用度数测量的角度。

### **ATan** (Value As Double) As Double

计算三角反正切函数。 返回用弧度测量的角度。

### **ATan2** (Y As Double, X As Double) As Double

计算三角反正切函数。 返回用弧度测量的角度。

### **ATan2D** (Y As Double, X As Double) As Double

计算三角反正切函数。 返回用度数测量的角度。

### **ATanD** (Value As Double) As Double

计算三角反正切函数。 返回用度数测量的角度。

### **BytesToString** (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String

将给定的字节数组解码为字符串。

数据 - 字节数组。

StartOffset - 要读取的第一个字节。

长度 - 要读取的字节数。

CharSet - 字符集的名称。

例子：

```
Dim s As String
s = BytesToString(Buffer, 0, Buffer.Length, "UTF-8")
```

### **CallSub** (Component As Object, Sub As String) As Object

调用给定的子程序。 CallSub 可用于调用属于不同模块的子程序。

但是，只有在其他模块没有暂停时才会调用子程序。 在这种情况下，将返回一个空字符串。

您可以使用 **IsPaused** 来测试模块是否已暂停。

这意味着一个活动不能调用不同活动的子程序。 因为其他活动肯定会暂停。

**CallSub** 允许活动调用子程序或服务调用活动子程序。

请注意，不能调用代码模块的子程序。

**CallSub** 也可以用于调用当前模块中的子程序。 在这种情况下，将 **Me** 作为组件传递。

例子：

```
CallSub(Main, "RefreshData")
```

### **CallSub2** (Component As Object, Sub As String, Argument As Object) As Object

类似于 **CallSub**。 使用单个参数调用 **sub**。

### **CallSub3** (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object

### **CallSubDelayed** (Component As Object, Sub As String)

**CallSubDelayed** 是 **StartActivity**、**StartService** 和 **CallSub** 的组合。

与仅适用于当前正在运行的组件的 **CallSub** 不同，**CallSubDelayed** 将在需要时首先启动目标组件。

**CallSubDelayed** 也可以用于调用当前模块中的子程序。 不是直接调用这些子程序，而是将一条消息发送到消息队列。

处理消息时将调用子程序。 这在您想要在当前子程序“之后”执行某些操作（通常与 UI 事件相关）的情况下很有用。

请注意，如果您在整个应用程序处于后台（没有可见活动）时调用 **Activity**，则一旦目标活动恢复，子程序将被执行。

### **CallSubDelayed2** (Component As Object, Sub As String, Argument As Object)

类似于 **CallSubDelayed**。 使用单个参数调用子程序。

### **CallSubDelayed3** (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object)

类似于 **CallSubDelayed**。 使用两个参数调用子程序。

### Catch

在 Try 块中抛出的任何异常都将在 Catch 块中被捕获。  
调用 `LastException` 以获取捕获的异常。

语法:

```
Try
...
Catch
...
End Try
```

### `cE As Double`

e (自然对数底) 常数。

### `Ceil (Number As Double) As Double`

返回大于或等于指定数字且等于整数的最小双精度数。

### `CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String`

通过从数组中复制字符来创建一个新字符串。  
复制从 `StartOffset` 开始, 复制的字符数等于 `Length`。

### `Chr (UnicodeValue As Int) As Char`

返回由给定 unicode 值表示的字符。

### Continue

停止执行当前迭代并继续下一个迭代。

### `Cos (Radians As Double) As Double`

计算三角余弦函数。以弧度测量的角度。

### `CosD (Degrees As Double) As Double`

计算三角余弦函数。以度为单位测量的角度。

### `cPI As Double`

PI 常数。

### CreateMap

使用给定的键/值对创建一个 地图 Map。

语法是: `CreateMap (key1: value1, key2: value2, ...)`

例子:

```
Dim m As Map = CreateMap("January": 1, "February": 2)
```

### CRLF As String

换行符。Chr(10) 的值。

### Dim

声明一个变量。

语法:

声明一个变量:

**Dim** 变量名 [**As** 类型] [= 表达式]

默认类型是字符串。

声明多个变量。所有变量都将是指定的类型。

**Dim** [**Const**] 变量 1 [= 表达式], 变量 2 [= 表达式], ..., [**As** 类型]

请注意, 速记语法仅适用于 **Dim** 关键字。

例子: **Dim** a = 1, b = 2, c = 3 **As** Int

声明一个数组:

**Dim** 变量(排名 1, 排名 2, ...) [**As** 类型]

例子: **Dim** Days(7) **As** String

对于零长度数组, 可以省略实际排名。

### Exit

退出最内层循环。

请注意, 在 **Select** 块内退出将退出 **Select** 块。

### False As Boolean

### Floor (Number As Double) As Double

返回小于或等于指定数字且等于整数的最大双精度数。

### For

语法:

**For** 变量 = 值 1 To 值 2 [步长间隔]

...

**Next**

如果迭代器变量之前未声明, 它将是 Int 类型。

或者:

**For Each** 值 **As** 类型 **In** 汇集

...

**Next**

例子:

**For** i = 1 To 10

    Log(i) '将打印 1 到 10 (含)。

**Next**

**For Each** n **As** Int **In** Numbers '数组

    Sum = Sum + n

**Next**

请注意, 循环限制只会在第一次迭代之前计算一次。

### **GetType (object As Object) As String**

返回表示对象的 java 类型的字符串。

### **If**

单线:

```
If 条件 Then 真实-声明 [Else 虚假-声明]
```

多行:

```
If 条件 Then
    声明
Else If 条件 Then
    声明
...
Else
    声明
End If
```

### **IIf**

内联如果 Inline If - 如果 条件 为 真实, 则返回 真值, 否则返回 假值。 仅评估相关表达式。

```
IIf (Condition As Boolean, TrueValue As Object, FalseValue As Object)
```

### **Is**

测试对象是否属于给定类型。

请注意, 当数字转换为对象时, 它可能会将其类型更改为不同类型的数字 (例如, Byte 可能会转换为 Int)。

例子:

```
For Each v As View in Page1.RootPanel.GetAllViewsRecursive
    If v Is Button Then
        Dim b As Button = v
        b.Color = Colors.Blue
    End If
Next
```

### **IsNumber (Text As String) As Boolean**

测试指定的字符串是否可以安全地解析为数字。

### **LoadBitmap (Dir As String, FileName As String) As Bitmap**

加载位图。

请注意, Android 文件系统区分大小写。

如果图像尺寸很大, 您应该考虑使用 **LoadBitmapSample**。

实际文件大小无关紧要, 因为图像通常是压缩存储的。

例子:

```
Activity.SetBackgroundImage(LoadBitmap(File.DirAssets, "SomeFile.jpg"))
```

🔓 **LoadBitmapResize** (Dir As String, FileName As String, Width As Int, Height As Int, KeepAspectRatio As Boolean) As Bitmap

加载位图并设置其大小。

位图比例将与设备比例相同。

与需要将容器 Gravity 设置为 FILL 的 LoadBitmapSample 不同，当 Gravity 设置为 CENTER 时，LoadBitmapResize 提供更好的结果。

例子：

```
Dim bd As BitmapDrawable = Activity.SetBackgroundImage(LoadBitmapResize(File.DirAssets, "SomeFile.jpg", 100%, 100%, True))
bd.Gravity = Gravity.CENTER
```

或者：

```
Activity.SetBackgroundImage(LoadBitmapResize(File.DirAssets, "SomeFile.jpg", 100%, 100%, True)).Gravity = Gravity.CENTER
```

🔓 **LoadBitmapSample** (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap

加载位图。

如果 MaxWidth 或 MaxHeight 小于位图尺寸，解码器将对位图进行二次采样。

这可以在加载大图像时节省大量内存。

例子：

```
Panel1.SetBackgroundImage(LoadBitmapSample(File.DirAssets, "SomeFile.jpg", Panel1.Width, Panel1.Height))
```

🔓 **Log** (Message As String)

记录一条消息。 可以在 Logs 选项卡中查看日志。

🔓 **Logarithm** (Number As Double, Base As Double) As Double

🔓 **LogColor** (Message As String, Color As Int)

记录一条消息。 该消息将以指定的颜色显示在 IDE 中。

🔓 **Max** (Number1 As Double, Number2 As Double) As Double

返回两个数字之间较大的数字。

🔓 **Me As Object**

对于类：返回对当前实例的引用。

对于活动和服务：返回对可与 CallSub, CallSubDelayed 和 SubExists 关键字一起使用的对象的引用。

不能在代码模块中使用。

🔓 **Min** (Number1 As Double, Number2 As Double) As Double

返回两个数字之间较小的数字。

### **Not** (Value As Boolean) As Boolean

反转给定布尔值的值。

### **Null** As Object

### **NumberFormat** (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String

将指定的数字转换为字符串。

该字符串将至少包含最小整数 Minimum Integers 和最多最大分数数字 Maximum Fractions。

例子：

```
Log(NumberFormat(12345.6789, 0, 2)) '"12,345.68"
Log(NumberFormat(1, 3, 0)) '"001"
```

### **NumberFormat2** (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String

将指定的数字转换为字符串。

该字符串将至少包含最小整数 Minimum Integers，最多最大分数数字 Maximum Fractions 和至少最小分数数字 Minimum Fractions。

GroupingUsed - 确定是否对每三个整数进行分组。

例子：

```
Log(NumberFormat2(12345.67, 0, 3, 3, false)) '"12345.670"
```

### **Power** (Base As Double, Exponent As Double) As Double

返回基值 Base 的指数幂 Exponent。

### **QUOTE** As String

引号字符 “。Chr(34) 的值。

### **Regex** As Regex

Regular expressions related methods.

### **Return**

从当前子返回并可选择返回给定值。

语法：Return [值]

### **Rnd** (Min As Int, Max As Int) As Int

返回 Min（包括）和 Max（不包括）之间的随机整数。

### **RndSeed** (Seed As Long)

设置随机种子值。

此方法可用于调试，因为它允许您每次都获得相同的结果。

### **Round** (Number As Double) As Long

返回与给定数字最接近的长数字。



### Round2 (Number As Double, DecimalPlaces As Int) As Double

舍入给定的数字并保留指定的小数位数。

### Select

将单个值与多个值进行比较。

例子：

```
Dim value As Int
value = 7
Select value
    Case 1
        Log("One")
    Case 2, 4, 6, 8
        Log("Even")
    Case 3, 5, 7, 9
        Log("Odd larger than one")
    Case Else
        Log("Larger than 9")
End Select
```

### Sender As Object

返回引发事件的对象。

仅在事件子内部有效。

例子：

```
Sub Button_Click
    Dim b As Button
    b = Sender
    b.Text = "I've been clicked"
End Sub
```

### Sin (Radians As Double) As Double

计算三角正弦函数。 以弧度测量的角度。

### SinD (Degrees As Double) As Double

计算三角正弦函数。 以度为单位测量的角度。

### Sleep (Value As Double) As Double

暂停当前子程序执行并在指定时间后恢复它。

### SmartStringFormatter (Format As String, Value As Object) As String

智能字符串文字使用的内部关键字。

### Sqrt (Value As Double) As Double

返回正平方根。

### Sub

用参数和返回类型声明一个子程序 sub。

语法：子程序名称 [(参数列表)] [作为返回类型]

参数包括名称和类型。

不应包括数组维度的长度。

例子：

```
Sub MySub (FirstName As String, LastName As String, Age As Int, OtherValues() As Double) As Boolean
```

```
...
```

```
End Sub
```

在此示例中，OtherValues 是一维数组。

返回类型声明与其他声明不同，因为数组括号跟随类型而不是名称（在这种情况下不存在）。

### SubExists (Object As Object, Sub As String) As Boolean

测试对象是否包含指定的方法。

如果对象未初始化或不是用户类的实例，则返回 false。

### TAB As String

制表符。

### Tan (Radians As Double) As Double

计算三角正切函数。 以弧度测量的角度。

### TanD (Degrees As Double) As Double

计算三角正切函数。 以度为单位测量的角度。

### True As Boolean

### Try

在 Try 块中抛出的任何异常都将在 Catch 块中被捕获。

调用 LastException 以获取捕获的异常。

语法：

```
Try
```

```
...
```

```
Catch
```

```
...
```

```
End Try
```

### Type

声明一个结构。

只能在 Sub Globals 或 Sub Process\_Globals 中使用。

句法:

键入类型名称 (字段 1, 字段 2, ...)

字段包括名称和类型。

例子:

```
Type MyType (Name As String, Items(10) As Int)
Dim a, b As MyType
a.Initialize
a.Items(2) = 123
```

### Until

循环直到条件为真。

句法:

Do Until 条件

...

Loop

### While

条件为真时循环。

语法:

Do While 条件

...

Loop

## 4.3 条件语句

B4X 中提供了不同的条件语句。

### 4.3.1 If - Then - Else

If-Then-Else 结构允许操作条件测试并根据测试结果执行不同的代码段。  
一般情况：

```
If test1 Then
    ' 代码1
Else If test2 Then
    ' 代码2
Else
    ' 代码3
End If
```

If-Then-Else 结构的工作原理如下：

1. 到达带有 **If** 关键字的行时，执行 **test1**。
2. 如果测试结果为 **True**，则执行 **代码 1**，直到带有 **Else If** 关键字的那一行。并跳转到 **End If** 关键字之后的行并继续。
3. 如果结果为 **False**，则执行 **test2**。
4. 如果测试结果为 **True**，则执行 **代码 2**，直到包含 **Else** 关键字的那一行。并跳转到 **End If** 关键字之后的行并继续。
5. 如果结果为 **False**，则执行 **代码 3** 并在 **End If** 关键字后面的行继续。

测试可以是任何类型的条件测试，有两种可能性 **True** 或 **False**。  
一些例子：

```
If b = 0 Then
    a = 0
End If
```

最简单的 If-Then 结构。

```
If b = 0 Then a = 0
```

相同，但在一行中。

```
If b = 0 Then
    a = 0
Else
    a = 1
End If
```

最简单的 If-Then-Else 结构。

```
If b = 0 Then a = 0 Else a = 1
```

相同，但在一行中。

就个人而言，我更喜欢多行的结构，更好的可读性。  
几十年前 HP Basic 的一个旧习惯，这个 Basic 每行只接受一条指令。

注意： 以下两个区别：

B4X	VB
<b>Else If</b>	<b>ElseIf</b>

在 B4X 中，**Else** 和 **If** 之间有一个空白字符。

一些用户尝试使用这种表示法：

```
If b = 0 Then a = 0 : c = 1
```

B4X 和 VB 之间有很大的不同会导致错误：

上述语句等价于：

B4X	VB
<b>If b = 0 Then</b> a = 0 <b>End If</b> c = 1	<b>If b = 0 Then</b> a = 0 c = 1 <b>End If</b>

以上一行中的冒号字符 ‘ : ’ 在 B4X 中视为 CarriageReturn CR 字符。

此结构会引发错误。

```
Sub Plus1 : x = x + 1 : End Sub
```

您不能在同一行有一个 Sub 声明和 End Sub。

#### 4.3.1.1 布尔求值顺序

在这个例子中：

```
If InitVar2(Var1) and Var1 > Var2 then ....
```

如果 InitVar2(Var1) 返回 false 是停止评估还是没有规则？

它从左到右运行，并在确定结果后立即停止（短路评估）。

这个非常重要。

它允许编写代码，例如：

```
If i < List.Size And List.Get(i) = "abc" Then
```

### 4.3.2 IIf 内联如果

IIf - 内联如果 Inline If, 也称为三元如果 *ternary if*, 因为它是具有三个参数的运算符。

```
Label1.Text = IIf(EditText1.Text <> "", EditText1.Text, "Please enter value")
```

IIf 基本上相当于这个子:

```
Sub PseudoIIf (Condition As Boolean, TrueValue As Object, FalseValue As Object) As Object
    If Condition = True Then Return TrueValue Else Return FalseValue
End Sub
```

与这个 sub 不同, IIf 关键字将只计算相关表达式。这意味着此代码将正常工作:

```
Return IIf(List1.Size > 0, List1.Get(0), "List is empty")
```

(还有一个与返回类型有关的细微差别。如果使用新的 **As** 方法显式设置它, 编译器将避免将值转换为 Object 并返回到目标类型。这仅在非常紧凑和长循环中有意义)。

### 4.3.3 Select - Case

Select - Case 结构允许将 **TestExpression** 与其他表达式进行比较，并根据 **TestExpression** 和表达式之间的匹配执行不同的代码段。

一般情况：

```
Select TestExpression
  Case ExpressionList1
    ' code1
  Case ExpressionList2
    ' code2
  Case Else
    ' code3
End Select
```

**TestExpression** 是要测试的表达式。

**ExpressionList1** 是要与 **TestExpression** 进行比较的表达式列表

**ExpressionList2** 是要与 **TestExpression** 进行比较的另一个表达式列表

Select - Case 结构的工作原理如下：

1. 评估 **TestExpression**。
2. 如果 **ExpressionList1** 中的一个元素与 **TestExpression** 匹配，则执行 **code1** 并在 **End Select** 关键字之后的行继续。
3. 如果 **ExpressionList2** 中的一个元素与 **TestExpression** 匹配，则执行 **code2** 并继续 **End Select** 关键字之后的行。
4. 对于没有匹配的表达式，**TestExpression** 执行 **code3** 并在 **End Select** 关键字之后的行继续。

**TestExpression** 可以是任何表达式或值。

**ExpressionList1** 是任何表达式或值的列表。

例子：

```
Select Value
  Case 1, 2, 3, 4
```

Value 变量是一个数值。

```
Select a + b
  Case 12, 24
```

**TestExpression** 是 a + b 的总和

```
Select Txt.CharAt(Index)
  Case "A", "B", "C"
```

**TestExpression** 是给定索引处的字符。

```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
  Select Action
    Case Activity.ACTION_DOWN

    Case Activity.ACTION_MOVE

    Case Activity.ACTION_UP

  End Select
End Sub
```

注意。 以下两者之间的差异：

B4X

```
Select Value  
  Case 1, 2, 3, 4, 8, 9, 10
```

VB

```
Select Case Value  
  Case 1 To 4, 8 To 9
```

在 VB 中，关键字 `Case` 被添加在 `Select` 关键字之后。

VB 接受 `Case 1 To 4`，这在 B4X 中没有实现。



## 4.4 循环结构

B4X 中提供了不同的循环结构。

### 4.4.1 For - Next

在 For-Next 循环中，代码块将被执行一定次数。

例子：

```
For i = n1 To n2 Step n3      i  增量变量
                              n1  初始值
      ' 代码块                n2  终值
                              n3  步
Next
```

For - Next 循环的工作原理如下：

1. 一开始，增量变量 **i** 等于初始值 **n1**。  
`i = n1`
2. 执行 **For** 和 **Next** 关键字之间的具体代码。
3. 到达 **Next** 时，增量变量 **i** 增加步长值 **n3**。  
`i = i + n3`。
4. 程序跳转回 **For**，比较增量变量 **i** 是否小于或等于最终值 **n2**。  
测试 `i <= n2`
5. 如果是，程序继续执行第 2 步，即 **For** 关键字之后的行。
6. 如果否，程序在 **Next** 关键字后面的行继续。

如果步长值等于“+1”，则不需要步长关键字。

```
For i = 0 To 10                For i = 0 To 10 Step 1
                                是相同
Next                            Next
```

步长变量可以是负数。

```
For i = n3 To 0 Step -1
Next
```

可以使用 **Exit** 关键字退出 For - Next 循环。

```
For i = 0 To 10                10 在本例中，如果变量 a 等于 0
      ' code
      If A = 0 Then Exit        然后退出循环。
      ' code
Next
```

**注意：** 以下两者之间的差异

B4X	VB
<code>Next</code>	<code>Next i</code>
<code>Exit</code>	<code>Exit For</code>

在 VB 中：

- 在 `Next` 关键字之后添加增量变量。
- 在 `Exit` 关键字之后指定循环类型。

### 4.4.2 For - Each

它是 For - Next 循环的变体。

例子：

<code>For Each n As Type In Array</code>	<code>n</code>	变量任何类型或对象
	<code>Type</code>	变量 <code>n</code> 的类型
<code>' 具体代码</code>	<code>Array</code>	值或对象数组
<code>Next</code>		

For - Each 循环的工作原理如下：

1. 一开始，`n` 获取 Array 中第一个元素的值。  
`n = Array(0)`
2. 执行 `For` 和 `Next` 关键字之间的具体代码。
3. 当到达 `Next` 时，程序检查 `n` 是否是数组中的最后一个元素。
4. 如果否，变量 `n` 获取数组中的下一个值并继续执行步骤 2，即 `For` 关键字之后的行。  
`n = Array(next)`
5. 如果是，程序在 `Next` 关键字后面的行继续。

For - Each 例子：

```
Private Numbers() As Int
Private Sum As Int

Numbers = Array As Int(1, 3, 5, 2, 9)

Sum = 0
For Each n As Int In Numbers
    Sum = Sum + n
Next
```

相同的示例，但带有 For - Next 循环：

```
Private Numbers() As Int
Private Sum As Int
Private i As Int

Numbers = Array As Int(1, 3, 5, 2, 9)

Sum = 0
For i = 0 To Numbers.Length - 1
    Sum = Sum + Numbers(i)
Next
```

此示例显示了 For - Each 循环的强大功能：

```
For Each lbl As Label In Activity
    lbl.TextSize = 20
Next
```

For - Next 循环的相同示例：

```
For i = 0 To Activity.NumberOfViews - 1
    Private v As View
    v = Activity.GetView(i)
    If v Is Label Then
        Private lbl As Label
        lbl = v
        lbl.TextSize = 20
    End If
Next
```

### 4.4.3 Do - Loop

存在几种配置：

```
Do While test          test 是任何表达式
    ' 代码              在 test 为 True 时执行代码
Loop
```

```
Do Until test          test 是任何表达式
    ' 代码              执行代码直到 test 为 True
Loop
```

Do While-Loop 循环的工作原理如下：

1. 一开始，对 **test** 进行评估。
2. 如果为 **True**，则执行代码
3. 如果 **False** 在 **Loop** 关键字之后的行继续。

Do Until-Loop 循环的工作原理如下：

1. 一开始，对 **test** 进行评估。
2. 如果为 **False**，则执行代码
3. 如果 **True** 在 **Loop** 关键字之后的行继续。

可以使用 **Exit** 关键字退出 Do-Loop 结构。

```
Do While test
    ' 代码
    If a = 0 Then Exit      如果 a = 0 则退出循环
    ' 代码
Loop
```

例子:

Do Until Loop:

```
Private i, n As Int

i = 0
Do Until i = 10
    ' 代码
    i = i + 1
Loop
```

Do While Loop:

```
Private i, n As Int

i = 0
Do While i < 10
    ' 代码
    i = i + 1
Loop
```

读取一个文本文件并填写一个列表:

```
Private lstText As List
Private line As String
Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirInternal, "test.txt"))
lstText.Initialize
line = tr.ReadLine
Do While line <> Null
    lstText.Add(line)
    line = tr.ReadLine
Loop

tr.Close
```

注意: 以下两者之间的差异

B4X	VB
<code>Exit</code>	<code>Exit Loop</code>

在 VB 中, 循环类型在 `Exit` 关键字之后指定。

VB 还接受以下循环, 这些循环在 B4X 中不受支持。

B4X	VB
<code>Do</code>	<code>Do</code>
<code>' 代码</code>	<code>' 代码</code>
<code>Loop While test</code>	<code>Loop Until test</code>

## 4.5 内联转换 As

As - 内联转换。允许从一种类型到另一种类型的内联转换。一些例子：

```
Dim Buttons As List = Array(Button1, Button2, Button3, Button4, Button5)
Dim s As String = Buttons.Get(2).As(B4XView).Text
Buttons.Get(2).As(B4XView).Text = "abc"
Dim j As String = "${
data: {
key1: value1,
complex_key2: {key: value2}
},
items: [0, 1, 2]
}"$

Dim parser As JSONParser
parser.Initialize(j)
Dim m As Map = parser.NextObject
Dim value1 As String = m.Get("data").As(Map).Get("key1")
Dim value2 As String = m.Get("data").As(Map).Get("complex_key2").As(Map).Get("key")
```

而且，对于 B4J：

```
Button1.As(JavaObject).RunMethod("setMouseTransparent", Array(True))
```

它也可以与数字一起使用，这在使用 JavaObject 调用外部 API 时特别有用，因为类型需要准确（对于 B4J）：

```
Log(Me.As(JavaObject).RunMethod("sum", Array((10).As(Float), (20).As(Double))))
'相当于:
Dim jme As JavaObject = Me
Dim f As Float = 10
Dim d As Double = 20
Log(jme.RunMethod("sum", Array(f, d)))

#if Java
public double sum(float n1, double n2) {
return n1 + n2;
}
#End If
```

## 4.6 子程序

子程序是一段代码。它可以是任意长度，并且具有独特的名称和定义的范围（以前面讨论的变量范围的方式）。在 B4X 代码中，子程序称为“Sub”，相当于其他编程语言中的过程，函数，方法和子程序。Sub 中的代码行从头到尾执行，如程序流程章节所述。

不建议使用包含大量代码的 Subs，它们的可读性会降低。

### 4.6.1 声明

子程序通过以下方式声明：

```
Sub CalcInterest (Capital As Double, Rate As Double) As Double
    Return Capital * Rate / 100
End Sub
```

它以关键字 **Sub** 开头，然后是子程序的名称，然后是参数列表，然后是返回类型，并以关键字 **End Sub** 结束。

子程序总是在模块的顶层声明，你不能将两个子程序嵌套在另一个里面。

### 4.6.2 调用子程序

当你想执行子程序中的代码行时，你只需写下子程序的名称。

例如：

```
Interest = CalcInterest(1234, 5.2)
```

Interest	子程序返回的值。
CalcInterest	子程序名称。
1235	传输到子程序的 Capital 值。
5.25	传输到子程序的 Rate 值。

### 4.6.3 从另一个模块调用子程序

在代码模块中声明的子程序例程可以从任何其他模块访问，但例程的名称必须以声明它的模块的名称为前缀。

示例：如果 CalcInterest 例程在模块 MyModule 中声明，则调用例程必须是：

```
Interest = MyModule.CalcInterest(1234, 5.2)
```

而不是：

```
Interest = CalcInterest(1234, 5.2)
```

#### 4.6.4 命名

基本上，您可以为子程序命名任何对变量合法的名称。 建议使用有意义的名称命名子程序，例如示例中的 **CalcInterest**，这样您可以通过阅读代码来了解它的作用。

您可以添加无限数量的子程序到程序中，但不允许在同一个模块中有两个同名的子程序。

#### 4.6.5 参数

参数 Parameters 可以传输到子程序。 该列表遵循子程序名称。 参数列表放在括号中。 参数类型应直接在列表中声明。

```
Sub CalcInterest (Capital As Double, Rate As Double) As Double
    Return Capital * Rate / 100
End Sub
```

在 B4X 中，参数是按值传输 by value 的，而不是按引用传输 by reference 的。

### 4.6.6 返回值

子程序可以返回一个值，这可以是任何对象。

返回值是使用 `Return` 关键字完成的。

返回值的类型添加在参数列表之后。

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
    Return Capital * Rate / 100
End Sub
```

您可以返回任何对象。

```
Sub InitList As List
    Private MyList As List
    MyList.Initialize

    For i = 0 To 10
        MyList.Add("Test" & i)
    Next
    Return MyList
End Sub
```

如果要返回一个数组，则需要在对象类型的末尾添加一个括号。

```
Sub StringArray As String ()
    Public strArr(2) As String
    strArr(0) = "Hello"
    strArr(1) = "world!"
    Return strArr
End Sub
```

如果要返回多维数组，则需要为每个补充维度添加逗号。  
一个逗号表示二维数组。

```
Sub StringMatrix As String (,)
    Public strMatrix(2,2) As String
    strMatrix(1,1) = "Hello world!"
    Return strMatrix
End Sub
```



## 4.7 可恢复子程序


可恢复子程序 (Resumable Subs) 是 B4A v7.00 / B4i v4.00 / B4J v5.50 中添加的一项新功能。它大大简化了异步任务的处理。

(此功能是无堆栈[协程](#)的变体。)

您可以在[论坛](#)中找到更多示例。

可恢复子程序的特殊功能是它们可以暂停，而无需暂停执行线程，稍后再恢复。程序不会等待可恢复子程序继续。其他事件将照常引发。

任何具有一个或多个 Sleep 或 Wait For 调用的子程序都是可恢复子程序。IDE 在子程序声明旁边显示此指示符：

```
Private Sub Countdown(Start As Int)   
    For i = Start To 0 Step -1  
        Label1.Text = i  
        Sleep(1000)  
    Next  
End Sub
```

### 4.7.1 Sleep

暂停当前子执行并在指定时间后恢复。

Sleep (Milliseconds As Int) Milliseconds, time delay in milliseconds.

例子：


```
Sleep(1000)
```

使用 Sleep 很简单：

```
Log(1)  
Sleep(1000)  
Log(2)
```

潜艇将暂停 1000 毫秒，然后恢复。

您可以调用 Sleep(0) 来实现最短的暂停。这可用于刷新 UI。它是 DoEvents 的一个很好的替代方案 (B4J 和 B4i 中不存在 DoEvents，在 B4A 中应避免使用)。

```
Sub VeryBusySub   
    For i = 1 To 10000000  
        '做点什么  
        If i Mod 1000 = 0 Then Sleep(0) '允许 UI 每 1000 次迭代刷新一次。  
    Next  
    Log("finished!")  
End Sub
```

### 4.7.2 Wait For

B4X 编程语言是事件驱动的。异步任务在后台运行，并在任务完成时引发事件。使用新的 Wait For 关键字，您可以在当前子程序中处理事件。

例如，此代码将等待 GoogleMap Ready 事件（B4J 示例）：

```
Sub AppStart (Form1 As Form, Args() As String)
    MainForm = Form1
    MainForm.RootPane.LoadLayout("1") 'Load the layout file.

    gmap.Initialize("gmap")
    Panel1.AddNode(gmap.AsPane, 0, 0, Panel1.Width, Panel1.Height)
    MainForm.Show
    Wait For gmap_Ready '<-----
    gmap.AddMarker(10, 10, "Marker")
End Sub
```

使用 FTP 的稍微复杂一点的示例：

列出远程文件夹中的所有文件，然后下载所有文件：

```
Sub DownloadFolder (ServerFolder As String)
    FTP.List(ServerFolder)
    Wait For FTP_ListCompleted (ServerPath As String, Success As Boolean, Folders() As
        FTPEntry, Files() As FTPEntry) '<----
    If Success Then
        For Each f As FTPEntry In Files
            FTP.DownloadFile(ServerPath & f.Name, False, File.DirApp, f.Name)
            Wait For FTP_DownloadCompleted (ServerPath2 As String, Success As Boolean) '<----
            Log($"File ${ServerPath2} downloaded. Success = ${Success}")
        Next
    End If
    Log("Finish")
End Sub
```

当调用 Wait For 关键字时，子程序会暂停，内部事件调度程序会在事件发生时将其恢复。如果事件从未发生，则子程序将永远不会恢复。程序仍将完全响应。

如果稍后使用同一事件调用 Wait For，则新的子程序实例将取代前一个。

假设我们要创建一个下载图像并将其设置为 `ImageView` 的子程序：

'坏例子。不要使用。

```
Sub DownloadImage(Link As String, iv As ImageView)
    Dim job As HttpJob
    job.Initialize("", Me) 'note that the name parameter is no longer needed.
    job.Download(Link)
    Wait For JobDone(job As HttpJob)
    If job.Success Then
        iv.SetImage (job.GetBitmap) 'replace with iv.Bitmap = job.GetBitmap in B4A / B4i
    End If
    job.Release
End Sub
```

如果我们调用它一次，它就会正常工作（更准确地说，如果我们在前一次调用完成之前不再调用它）。

如果我们像这样调用它：

```
DownloadImage("https://www.b4x.com/images3/android.png", ImageView1)
DownloadImage("https://www.b4x.com/images3/apple.png", ImageView2)
```

然后只会显示第二幅图像，因为第二次调用 `Wait For JobDone` 将覆盖前一个图像。

这将我们带到了 `Wait For` 的第二种变体。

为了解决这个问题，`Wait For` 可以根据事件发送者区分事件。

这是通过一个可选参数完成的：

*Wait For* (<sender>) <event signature>

例子：

'很好的例子。使用。

```
Sub DownloadImage(Link As String, iv As ImageView)
    Dim job As HttpJob
    job.Initialize("", Me) 'note that the name parameter is no longer needed.
    job.Download(Link)
    Wait For (job) JobDone(job As HttpJob)
    If job.Success Then
        iv.SetImage (job.GetBitmap) 'replace with iv.Bitmap = job.GetBitmap in B4A / B4i
    End If
    job.Release
End Sub
```

通过上述代码，每个可恢复的子实例将等待不同的事件，并且不会受到其他调用的影响。

不同之处在于 `Wait For` 行：

坏的：`Wait For JobDone(job As HttpJob)`

好的：`Wait For (job) JobDone(job As HttpJob)`

### 4.7.3 代码流

```
Sub S1
  Log("S1: A")
  S2
  Log("S1: B")
End Sub
```

```
Sub S2
  Log("S2: A")
  Sleep(0)
  Log("S2: B")
End Sub
```

输出为:



```
S1: A
S2: A
S1: B
S2: B
```

每当调用 `Sleep` 或 `Wait For` 时，当前子程序都会暂停。这相当于调用 `Return`。

#### 4.7.4 等待可恢复子程序任务完成

当一个子程序调用第二个可恢复子程序时，第一个子程序中的代码将在第一个 Sleep 或 Wait For 调用（在第二个子程序中）之后继续执行。

如果您想等待第二个子程序完成，那么您可以从第二个子程序引发一个事件并在第一个子程序中等待它：

```
Sub FirstSub   
    Log("FirstSub started")  
    SecondSub  
    Wait For SecondSub_Complete  
    Log("FirstSub completed")  
End Sub  
  
Sub SecondSub   
    Log("SecondSub started")  
    Sleep(1000)  
    Log("SecondSub completed")  
    CallSubDelayed(Me, "SecondSub_Complete")  
End Sub
```

日志：

```
FirstSub started  
SecondSub started  
SecondSub completed  
FirstSub completed
```

注意：

- 使用 CallSubDelayed 比使用 CallSub 更安全。如果第二个子程序从未暂停（例如，如果仅根据某些条件调用 Sleep），CallSub 将失败。
- 这里有一个假设，即 FirstSub 在完成之前不会再次被调用。

### 4.7.5 可恢复子程序返回值

可恢复子程序可以返回一个 *ResumableSub* 值。

例子：

```
Sub Button1_Click
    Sum(1, 2)
    Log("after sum")
End Sub

Sub Sum(a As Int, b As Int)
    Sleep(100) '这将导致代码流返回到父级
    Log(a + b)
End Sub
```

输出：

```
after sum
3
```

这就是为什么不能简单地返回一个值的原因。

**解决方案。**

可恢复的子程序可以返回一个名为 *ResumableSub* 的新类型。其他子程序可以使用此值等待子程序完成并获取所需的返回值。

```
Sub Button1_Click
    Wait For(Sum(1, 2)) Complete (Result As Int)
    Log("result: " & Result)
    Log("after sum")
End Sub

Sub Sum(a As Int, b As Int) As ResumableSub
    Sleep(100)
    Log(a + b)
    Return a + b
End Sub
```

输出：

```
3
result: 3
after sum
```

上述 Button1\_Click 代码等效于：

```
Sub Button1_Click
    Dim rs As ResumableSub = Sum(1, 2)
    Wait For(rs) Complete (Result As Int)
    Log("result: " & Result)
    Log("after sum")
End Sub
```

所需步骤如下：

1. 将 `As ResumableSub` 添加到可恢复子签名中。
2. 使用您想要返回的值调用 `Return`。
3. 在调用子程序中，使用 `Wait For (<sub here>) Complete (Result As <matching type>)` 调用可恢复子程序

注意事项和提示：

- 如果您不需要返回值但仍想等待可恢复子程序完成，则从可恢复子程序返回 `Null`，并将调用子程序中的类型设置为 `Object`。
- 多个子程序可以安全地调用可恢复子程序。完成事件将到达正确的父级。
- 您可以在其他模块中等待可恢复子程序（在 B4A 中，它仅与类相关）。
- 可以更改结果参数名称。

### 4.7.6 B4A 仅限 KeyPress 和 Wait For MsgBox2Async

在 B4A 中，通常会检查“返回”键，以防止用户无意中退出程序。

您可以使用此代码：

```
Sub Activity_KeyPress (KeyCode As Int) As Boolean '返回 True 以使用事件
    Select KeyCode
        Case KeyCodes.KEYCODE_BACK
            OpenMsgBox
            Return True
        Case Else
            Return False
    End Select
End Sub

Sub OpenMsgBox
    Private Answ As Int

    MsgBox2Async("Do you want to exit?", "E x i t", "Yes", "", "No", Null, False)
    Wait For MsgBox_Result (Answ As Int)
    If Answ = DialogResponse.POSITIVE Then
        Activity.Finish
    End If
End Sub
```



### 4.7.7 DoEvents 已弃用！

从 B4A v7.0 开始，DoEvents 调用将出现以下警告：

**DoEvents 已弃用。它可能导致稳定性问题。改用 Sleep(0)（如果确实需要）。**

DoEvents 的目的是允许在主线程繁忙时更新 UI。DoEvents 与模式对话框实现共享相同的实现，是一种低级实现。它访问进程消息队列并运行一些等待消息。

随着 Android 的发展，消息队列的处理变得更加复杂和脆弱。

弃用 DoEvents 的原因是：

1. 它是不稳定问题的主要来源。它可能导致难以调试的崩溃或 ANR（应用程序无响应）对话框。请注意，这也适用于模式对话框（例如 MsgBox 和 InputList）。
2. 有更好的方法可以让主线程保持空闲。例如，使用[异步 SQL 方法](#)而不是同步方法。
3. 它没有执行许多开发人员期望它执行的操作。由于它仅处理与 UI 相关的消息，因此大多数事件无法通过 DoEvents 调用引发。
4. 现在可以调用 Sleep 暂停当前子进程，并在处理完等待消息后恢复它。[Sleep 实现](#)与 DoEvents 完全不同。它不保留线程。而是在保留子进程状态的同时释放线程。  
与仅处理与 UI 相关的消息的 DoEvents 不同，使用 Sleep 将处理所有消息并引发其他事件。  
（请注意，使用 Wait For 等待事件比在循环中调用 Sleep 更好。）

话虽如此，DoEvents 仍然存在，现有应用程序将完全像以前一样工作。

### 4.7.8 对话框

模态对话框 = 保持主线程直到对话框关闭的对话框。

如上所述，模态对话框与 `DoEvents` 共享相同的实现。因此建议改用新的异步对话框。  
使用 [Wait For](#) 实际上是一个简单的更改：

而不是：

```
Dim res As Int = MsgBox2("Delete?", "Title", "Yes", "Cancel", "No", Null)
If res = DialogResult.POSITIVE Then
    ...
End If
```

您应该使用：

```
Msgbox2Async("Delete?", "Title", "Yes", "Cancel", "No", Null, False)
Wait For MsgBox_Result (Result As Int)
If Result = DialogResult.POSITIVE Then
    ...
End If
```

*Wait For* 不会占用主线程。相反，它会保存当前子状态并释放它。当用户单击其中一个对话框按钮时，代码将恢复。

其他类似的新方法有：`MsgboxAsync`，`InputListAsync` 和 `InputMapAsync`。

除了 `MsgboxAsync` 之外，新方法还添加了一个新的 *可取消* 参数。如果该参数为真，则可以通过单击返回键或在对话框外部关闭对话框。这是旧方法的默认行为。

由于其他代码可以在异步对话框可见时运行，因此可能会同时出现多个对话框。

如果这种情况与您的应用相关，那么您应该在 `Wait For` 调用中设置发送方过滤器参数：

```
Dim sf As Object = MsgBox2Async("Delete?", "Title", "Yes", "Cancel", "No", Null, False)
Wait For (sf) MsgBox_Result (Result As Int)
If Result = DialogResult.POSITIVE Then
    ...
End If
```

这允许显示多条消息并且结果事件将被正确处理。

### 4.7.9 带有 Wait For 的 SQL

新的可恢复子功能使得处理大型数据集变得更简单，同时对程序响应能力的影响最小。

插入数据的新标准方式是：

```
For i = 1 To 1000
    SQL1.AddNonQueryToBatch("INSERT INTO table1 VALUES (?)", Array(Rnd(0, 100000)))
Next
Dim SenderFilter As Object = SQL1.ExecNonQueryBatch("SQL")
Wait For (SenderFilter) SQL_NonQueryComplete (Success As Boolean)
Log("NonQuery: " & Success)
```

步骤如下：

- 为每个应发出的命令调用 AddNonQueryToBatch。
- 使用 ExecNonQueryBatch 执行命令。这是一种异步方法。命令将在后台执行，完成后将引发 NonQueryComplete 事件。
- 此调用返回一个可用作发送方筛选器参数的对象。这很重要，因为可能有多个后台批处理执行正在运行。使用筛选器参数，在所有情况下，正确的 Wait For 调用都会捕获事件。
- 请注意，SQL1.ExecNonQueryBatch 在内部开始和结束事务。

#### 4.7.9.1 查询

在大多数情况下，查询速度会很快，因此应与 SQL1.ExecQuery2 同步发出。但是，如果查询速度很慢，则应切换到 SQL1.ExecQueryAsync：

```
Dim SenderFilter As Object = SQL1.ExecQueryAsync("SQL", "SELECT * FROM table1", Null)
Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
If Success Then
    Do While rs.NextRow
        Log(rs.GetInt2(0))
    Loop
    rs.Close
Else
    Log(LastException)
End If
```

与前一种情况一样，ExecQueryAsync 方法返回一个用作发送方过滤器参数的对象。

提示：

1. B4A 中的 ResultSet 类型扩展了 Cursor 类型。如果愿意，您可以将其更改为 Cursor。使用 ResultSet 的优点是它与 B4J 和 B4i 兼容。
2. 如果查询返回的行数很大，则 Do While 循环在调试模式下会很慢。您可以将其放在不同的子项目中并清理项目（Ctrl + P），以使其更快：

```
Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
If Success Then
    WorkWithResultSet(rs)
Else
    Log(LastException)
End If
End Sub

Private Sub WorkWithResultSet(rs As ResultSet)
    Do While rs.NextRow
        Log(rs.GetInt2(0))
    Loop
    rs.Close
End Sub
```

这与可恢复子系统中当前已禁用的调试器优化有关。  
在发布模式下，两种解决方案的性能将相同。

#### 4.7.9.2 B4J

- 需要 jSQL v1.50+ (<https://www.b4x.com/android/forum/threads/updates-to-internal-libraries.48274/#post-503552>).
- 建议将日志模式设置为 WAL: <https://www.b4x.com/android/forum/t...ent-access-to-sqlite-databases.39904/#content>

#### 4.7.10 注意事项和提示

- 在大多数情况下，可恢复子程序在发布模式下的性能开销应该微不足道。在调试模式下，开销可能会更大。（如果这成为一个问题，则将代码中运行缓慢的部分移至从可恢复子程序调用的其他子程序。）
- 等待事件处理程序先于常规事件处理程序。
- 可恢复子程序不会创建额外的线程。代码由主线程或服务器解决方案中的处理程序线程执行。

## 4.8 事件

在面向对象编程中，我们有可以对不同用户操作（称为事件）做出反应的对象。对象可以引发的事件的数量和类型取决于对象的类型。

### 4.8.1 B4A

在安卓中，用户界面对象被称为“视图”。

不同视图的事件摘要:

[illegible]

最常见的事件是：

- **Click** 当用户点击视图时引发事件。  
例子：  

```
Sub Button1_Click
    ' 您的代码
End Sub
```
- **LongClick** 当用户点击视图并按住一段时间时引发事件。  
例子：  

```
Sub Button1_LongClick
    ' 您的代码
End Sub
```
- **Touch (Action As Int, X As Float, Y As Float)**  
当用户触摸屏幕时引发事件。

处理三种不同的操作：

- Activity.ACTION\_DOWN, 用户触摸屏幕。
- Activity.ACTION\_MOVE, 用户移动手指而不离开屏幕。
- Activity.ACTION\_UP, 用户离开屏幕。

给出了手指位置的 X 和 Y 坐标。

例子：

```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Activity.ACTION_DOWN
        ' 您的 DOWN 操作代码
    Case Activity.ACTION_MOVE
        ' 您的 MOVE 行动代码
    Case Activity.ACTION_UP
        ' 您的 UP 行动代码
    End Select
End Sub
```

- **CheckChanged (Checked As Boolean)**  
当用户点击 CheckBox 或 RadioButton 时引发的事件  
如果视图被选中，则 Checked 等于 True，如果未选中，则 Checked 等于 False。

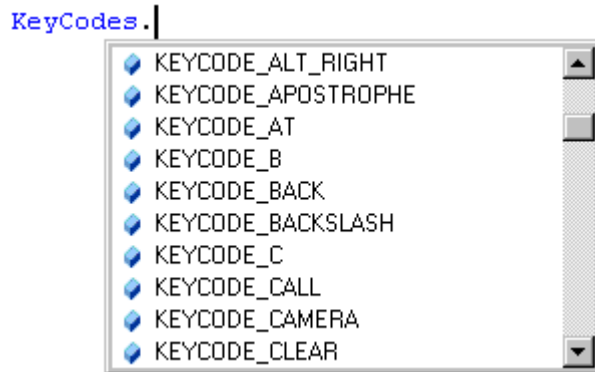
例子：

```
Sub CheckBox1_CheckedChange(Checked As Boolean)
    If Checked = True Then
        ' 您的代码（如果已检查）
    Else
        ' 您的代码（如果未检查）
    End If
End Sub
```

- **KeyPress (KeyCode As Int) As Boolean**

当用户按下物理或虚拟按键时引发事件。

KeyCode 是所按下按键的代码，您可以使用 KeyCodes 关键字获取它们。



事件可以返回：

- True, 事件已被“消耗”，操作系统认为事件已执行，不会采取进一步行动。
- False, 事件未被消耗，并传输到系统进行进一步操作。

例子：

```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
    Private Answ As Int
    Private Txt As String

    If KeyCode = KeyCodes.KEYCODE_BACK Then ' 检查 KeyCode 是否为 Back Key
        Txt = "Do you really want to quit the program ?"
        Answ = MsgBox2(Txt,"A T T E N T I O N","Yes","","No",Null)' 消息框
        If Answ = DialogResponse.POSITIVE Then ' 如果返回值为 Yes, 则
            Return False ' 返回 = False 事件 不会被消耗
        Else ' 我们离开程序
            Return True ' 返回 = True 事件 将被消耗以避免
        End If ' 退出程序
    End If
End Sub
```

#### 4.8.2 B4i

在 iOS 中，用户界面对象被称为“视图”。

不同视图的事件摘要:

[illegible]



最常见的事件是：

- **Click**            当用户点击视图时引发事件。  
例子：  
`Private Sub Button1_Click`  
    ' 您的代码  
`End Sub`
- **LongClick**      当用户点击视图并按住一段时间时引发事件。  
例子：  
`Private Sub Button1_LongClick`  
    ' 您的代码  
`End Sub`
- **Touch (Action As Int, X As Float, Y As Float)**  
当用户触摸屏幕上的面板时引发事件。

处理三种不同的操作：

- `Panel.ACTION_DOWN`,        用户触摸屏幕。
- `Panel.ACTION_MOVE`,        用户移动手指而不离开屏幕。
- `Panel.ACTION_UP`,         用户离开屏幕。

手指位置的 X 和 Y 坐标以点为单位给出，而不是以像素为单位。

例子：

```
Private Sub Panel_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Panel.ACTION_DOWN
        ' 您的 DOWN 操作代码
    Case Panel.ACTION_MOVE
        ' 您的 MOVE 行动代码
    Case Panel.ACTION_UP
        ' 您的 UP 行动代码
    End Select
End Sub
```

### 4.8.3 B4J

用户界面对象在 Java 中被称为“节点”(Nodes)。

不同节点的事件摘要:

[illegible]

最常见的事件是：

- **Action**            当用户点击节点（按钮或文本字段）时引发事件。  
例子：  

```
Private Sub Button1_Action
    ' 您的代码
End Sub
```
- **FocusChanged** (HasFocus As Boolean)    当节点获得或失去焦点时引发事件。  
例子：  

```
Private Sub TextField1_FocusChanged (HasFocus As Boolean)
    ' 您的代码
End Sub
```
- **MouseClicked** (EventData As MouseEvent)  
当用户点击节点时引发事件。  
例子：  

```
Private Sub Panel1_MouseClicked (EventData As MouseEvent)
    ' 您的代码
End Sub
```
- **MouseDragged** (EventData As MouseEvent)  
当用户拖动节点（按下按钮移动）时引发事件。  
类似于 B4A Touch 事件中的 ACTION\_MOVE。  
例子：  

```
Private Sub Panel1_MouseDragged (EventData As MouseEvent)
    ' 您的代码
End Sub
```
- **MouseEntered** (EventData As MouseEvent)  
当用户进入节点时引发的事件。  
例子：  

```
Private Sub Panel1_MouseEntered (EventData As MouseEvent)
    ' 您的代码
End Sub
```
- **MouseExited** (EventData As MouseEvent)  
当用户退出节点时引发的事件。  
例子：  

```
Private Sub Panel1_MouseExited (EventData As MouseEvent)
    ' 您的代码
End Sub
```
- **MouseMoved** (EventData As MouseEvent)  
当用户移动到节点上（没有按下按钮）时引发事件。  
例子：  

```
Private Sub Panel1_MouseMoved (EventData As MouseEvent)
    ' 您的代码
End Sub
```

- **MousePressed (EventData As MouseEvent)**

当用户按下节点时引发事件。

类似于 B4A Touch 事件中的 ACTION\_DOWN。

例子：

```
Private Sub Panel1_MousePressed (EventData As MouseEvent)
    ' 您的代码
End Sub
```

- **MouseReleased (EventData As MouseEvent)**

当用户释放节点时引发事件。

类似于 B4A Touch 事件中的 ACTION\_UP。

例子：

```
Private Sub Panel1_MouseReleased (EventData As MouseEvent)
    ' 您的代码
End Sub
```

- **MouseEvent**

MouseEvent 对象中包含的数据：

- |                          |                            |
|--------------------------|----------------------------|
| • ClickCount             | 返回与此事件相关的点击次数。             |
| • Consume                | 消耗当前事件并阻止其被节点父级处理。         |
| • MiddleButtonDown       | 如果中间按钮当前处于按下状态，则返回 true。   |
| • MiddleButtonPressed    | 如果中间按钮负责引发当前点击事件，则返回 true。 |
| • PrimaryButtonDown      | 如果主按钮当前处于按下状态，则返回 true。    |
| • PrimaryButtonPressed   | 如果主按钮负责引发当前点击事件，则返回 true。  |
| • SecondaryButtonDown    | 如果辅助按钮当前处于按下状态，则返回 true。   |
| • SecondaryButtonPressed | 如果辅助按钮负责引发当前点击事件，则返回 true。 |
| • X                      | 返回与节点边界相关的 X 坐标。           |
| • Y                      | 返回与节点边界相关的 Y 坐标。           |

例子：

```
Private Sub pnlMain_MouseMoved (EventData As MouseEvent)
    Private x, y As Int

    If EventData.MiddleButtonPressed = True Then
        x = EventData.X
        y = EventData.Y
        ' 其他代码
    End If
End Sub
```

- **Touch (Action As Int, X As Float, Y As Float)**  
当用户“触摸”屏幕时引发事件。  
此事件类似于 B4A 和 B4i 中的触摸事件。

处理三种不同的操作：

- Panel1.TOUCH\_ACTION\_DOWN, 用户触摸屏幕。
- Panel1.TOUCH\_ACTION\_MOVE, 用户移动手指而不离开屏幕。
- Panel1.TOUCH\_ACTION\_UP, 用户离开屏幕。

给出了鼠标光标位置的 X 和 Y 坐标。

例子：

```
Sub Panel1_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Panel1.TOUCH_ACTION_DOWN
        ' 您的 DOWN 行动代码
    Case Panel1.TOUCH_ACTION_MOVE
        ' 您的 MOVE 行动代码
    Case Panel1.TOUCH_ACTION_UP
        ' 您的 UP 行动代码
    End Select
End Sub
```

或

```
Sub Panel1_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case 0 'DOWN
        ' 您的 DOWN 行动代码
    Case 2 'MOVE
        ' 您的 MOVE 行动代码
    Case 1 'UP
        ' 您的 UP 行动代码
    End Select
End Sub
```

## 4.8.4 B4R

在 B4R 中，只有 Pin 和 [Timer](#) 对象会引发事件：

- Pin  
StateChanged (State As Boolean) 当引脚改变其状态时引发事件。

例子：

```
Sub Pin1_StateChanged(State As Boolean)
    ' 您的代码
End Sub
```

- Timer  
Tick 每隔给定间隔引发事件

例子：

```
Private Timer1 As Timer

Timer1.Initialize("Timer1_Tick",1000)

Sub Timer1_Tick
    ' 您的代码
End Sub
```

请注意，B4R 中的初始化方法与其他 B4X 产品不同。

您必须声明完整的子名称，如 "Timer1\_Tick"，而不是像其他产品中那样声明 "Timer1"。

4.8.5 用户界面摘要

“标准”用户界面对象。  
这显示了三个操作系统之间的差异。  
一些不作为标准对象存在的视图/节点可以作为 CustomViews 存在于其他操作系统中。您应该查看论坛。

View / node	B4A	B4i	B4J
Activity			
Button			
CheckBox			
EditText			
HorizontalScrollView			
ImageView			
Label			
ListView			
Panel			
RadioButton			
ScrollView			
SeekBar			
Spinner			
TabHost			
ToggleButton			
WebView			
TextField			
TextView			
ScrollView 与 B4A 2D 不同			
Slider			
Picker			
Stepper			
Switch			
SegmentedControl			
Canvas 独立节点			
ChoiceBox			
ComboBox			
Pane 类似于 B4A 和 B4i 中的面板			
ScrollPane similar to ScrollView			
TabPane			
TextArea			

对于跨平台项目，您可以查看 [B4X 跨平台项目](#)手册和更具体的[第 4 章。兼容性 B4A B4i B4J XUI](#)。

## 4.9 库

库为 B4X 添加了更多对象和功能。

其中一些库随 B4X 产品一起提供，是标准开发系统的一部分。

其他库通常由用户开发，可以下载（仅限注册用户）以向 B4X 开发环境添加补充功能。

当您需要一个库时，您必须：

- 如果您已经有该库，请在 Libs 选项卡中检查它。
- 对于其他库，请检查它是否是最新版本。

您可以在文档页面 [B4A](#)，[B4i](#)，[B4J](#)，[B4R](#) 中查看版本

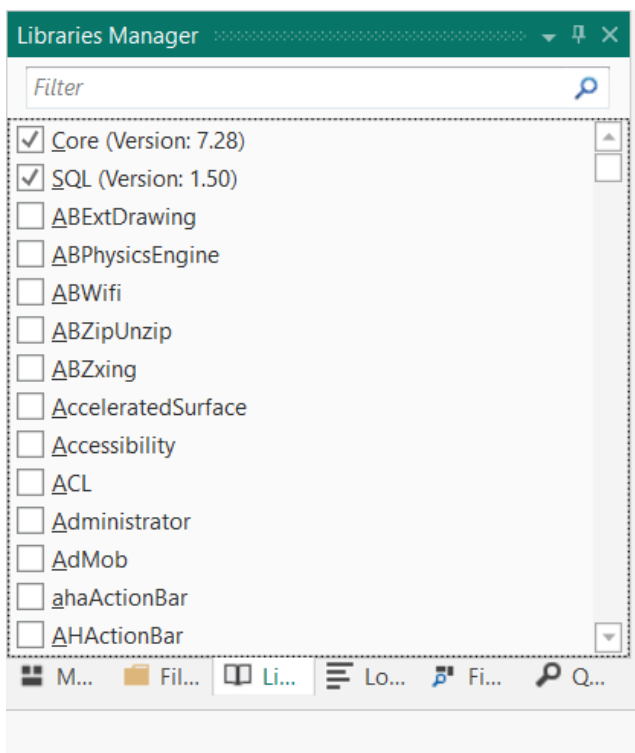
或在论坛中的 [Libraries Google 表格](#) 中查看。

要查找库文件，请在您的互联网浏览器中使用类似

<http://www.b4x.com/search?query=betterdialogs+library>

的查询。

- 如果是，则检查列表中的库以将其选中。

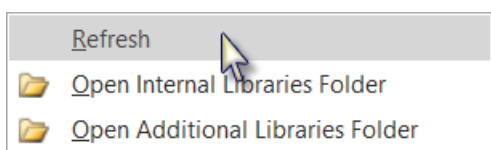


- 如果不是，请下载库，解压并将

<库名称>.jar 和 <库名称>.xml 文件复制到给定产品的附加库文件夹。

如果是 [B4XLibrary](#)，请将 <库名称>.b4xlib 文件复制到附加库\B4X 文件夹。

- 右键单击 Lib 区域，然后单击 **Refresh** 并选中列表中的库以将其选中。





### 4.9.1 标准库

标准 B4X 库保存在 B4X 程序文件夹中的 Libraries 文件夹中。

通常位于：

C:\Program Files\Anywhere Software\B4A\Libraries

C:\Program Files\Anywhere Software\B4i\Libraries

C:\Program Files\Anywhere Software\B4J\Libraries

C:\Program Files\Anywhere Software\B4R\Libraries

### 4.9.2 附加库文件夹

附加库由两个文件组成：*xxx.jar* 和 *xxx.xml* 文件。

B4X 库只有一个文件 *xxx.b4xlib*。

对于附加库，需要设置一个特殊文件夹以将它们保存在其他地方。

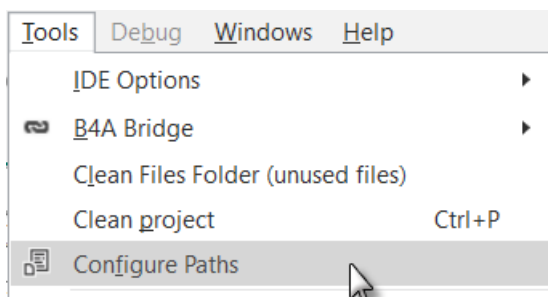
此文件夹必须具有以下结构：

▼ AdditionalLibraries	
B4A	
B4i	B4A 附加库文件夹。
B4J	B4i 附加库文件夹。
B4R	B4J 附加库文件夹。
B4X	B4R 附加库文件夹。
Snippets	<a href="#">B4X 库</a> 文件夹。
B4XlibXMLFiles	<a href="#">代码片段</a> 文件夹。B4X 文件夹中的子文件夹。
	B4X 库 XML 文件的文件夹。

每个产品一个子文件夹：B4A，B4i，B4J，B4R 以及用于 B4X 库的另一个 B4X。

当您安装 B4X 产品的新版本时，所有标准库都会自动更新，但附加库不会包含在内。特殊文件夹的优点是您无需关心它们，因为安装新版本的 B4X 时此文件夹不会受到影响。

附加库不会随新版本的 B4X 系统更新。



当 IDE 启动时，它首先在 B4X 的 Libraries 文件夹中查找可用的库，然后在附加库文件夹中查找。

要设置特殊的附加库文件夹，请单击 IDE 菜单上的“工具/配置路径”。

在我的系统中，我添加了一个 B4XlibXMLFiles 文件夹用于存放 XML 帮助文件。

标准库和附加库都有一个 XML 文件。B4X 库没有。

但是，如果您使用 [B4X 帮助查看器](#)，您会对这些帮助文件感兴趣（如果它们可用）。B4X 帮助查看器在 [B4X 帮助工具手册](#)中有说明。

您可以使用此工具为 b4xlib 库创建 xml 文件：[b4xlib - XML 生成](#)。

### 4.9.2.1 路径配置 B4A

**Paths Configuration**

javac.exe    
Usually found under C:\Program Files\Java\jdk1.8.x\_xx\bin

SDK Manager  Proxy Host:  Proxy Port:

android.jar    
Usually found under C:\android-sdk\platforms\android-x

Additional Libraries    
(optional) A folder where libraries will be searched for, in addition to the internal libraries folder. Make sure NOT to set it to the B4A libraries folder.

Shared Modules    
(optional) A folder where code modules will be searched for, in addition to the project folder.

输入文件夹名称并点击 .

### 4.9.2.2 路径配置 B4i

**Paths Configuration**

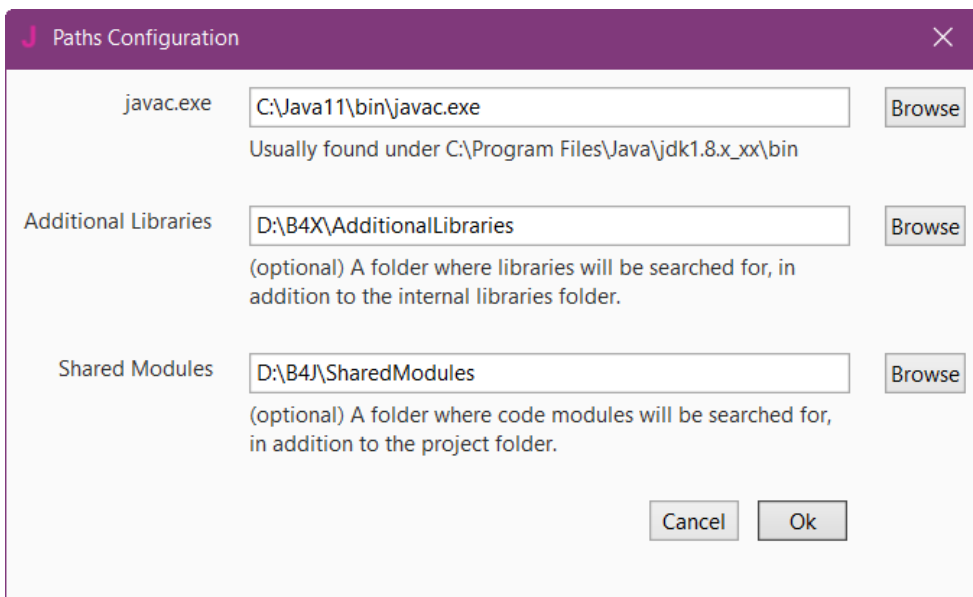
javac.exe    
Usually found under C:\Program Files\Java\jdk1.8.x\_xx\bin

Keys Folder    
A folder for the keys related files.

Additional Libraries    
(optional) A folder where libraries will be searched for, in addition to the internal libraries folder.

Shared Modules    
(optional) A folder where code modules will be searched for, in addition to the project folder.

### 4.9.2.3 路径配置 B4J

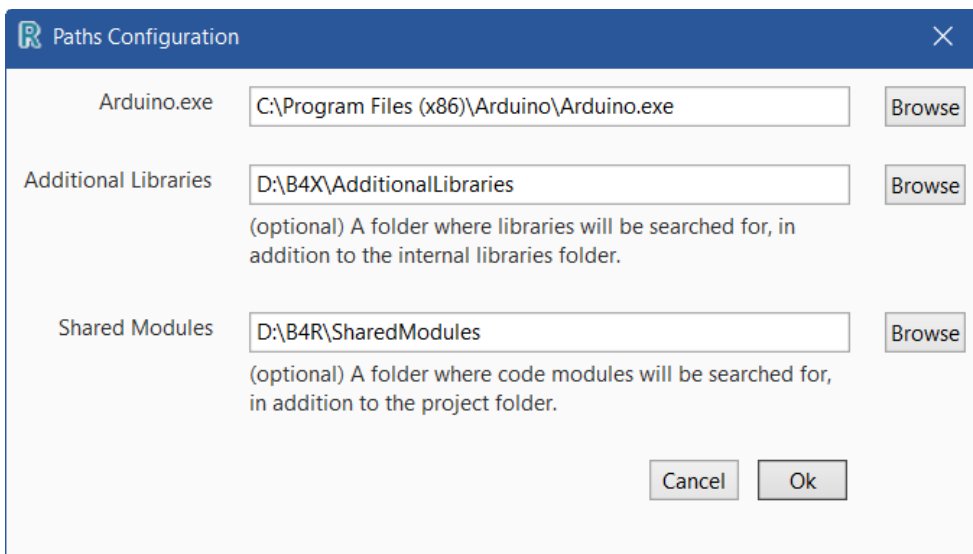


The B4J Paths Configuration dialog box has a purple title bar with the Java logo and the text "Paths Configuration". It contains three rows of configuration fields:

- javac.exe**: A text field containing "C:\Java11\bin\javac.exe" with a "Browse" button to its right. Below the field is the text: "Usually found under C:\Program Files\Java\jdk1.8.x\_xx\bin".
- Additional Libraries**: A text field containing "D:\B4X\AdditionalLibraries" with a "Browse" button to its right. Below the field is the text: "(optional) A folder where libraries will be searched for, in addition to the internal libraries folder."
- Shared Modules**: A text field containing "D:\B4J\SharedModules" with a "Browse" button to its right. Below the field is the text: "(optional) A folder where code modules will be searched for, in addition to the project folder."

At the bottom right, there are "Cancel" and "Ok" buttons.

### 4.9.2.4 路径配置 B4R



The B4R Paths Configuration dialog box has a blue title bar with the Arduino logo and the text "Paths Configuration". It contains three rows of configuration fields:

- Arduino.exe**: A text field containing "C:\Program Files (x86)\Arduino\Arduino.exe" with a "Browse" button to its right.
- Additional Libraries**: A text field containing "D:\B4X\AdditionalLibraries" with a "Browse" button to its right. Below the field is the text: "(optional) A folder where libraries will be searched for, in addition to the internal libraries folder."
- Shared Modules**: A text field containing "D:\B4R\SharedModules" with a "Browse" button to its right. Below the field is the text: "(optional) A folder where code modules will be searched for, in addition to the project folder."

At the bottom right, there are "Cancel" and "Ok" buttons.

### 4.9.3 B4X 库 \*.b4xlib

B4X 库是 B4A 8.80, B4i 5.50 和 B4J 7.00 中引入的跨平台库。

这些库包含跨平台类，无需编译为库。

B4XLibraries 在 [B4X 自定义视图手册](#) 中进行了说明。

B4X 库是一个简单的 zip 文件，具有以下结构：

- 代码模块。支持所有类型，包括活动和服务。
- 文件，包括布局文件。
- 可选清单文件，具有以下字段：
  - 版本
  - 作者
  - DependsOn（所需库的列表），支持的平台。字段可以在平台之间共享，也可以是特定于平台的。
- [代码片段](#) 文件夹，其中包含特定于库的代码片段。

文件和代码模块也可以是特定于平台的。




创建 B4X 库非常简单。您只需创建一个包含这些资源的 zip 文件。Zip 文件扩展名应为 b4xlib。就这样。

请注意，可以从 B4X 库中提取源代码。




B4X 库与“库”选项卡中的所有其他库一样出现。

示例：AnotherDatePicker.b4xlib

zip 文件结构：

 Files  
 AnotherDatePicker.bas  
 manifest.txt

Files 包含所有需要的文件，即示例中的三个布局文件。

 DatePicker.bal  
 DatePicker.bil  
 DatePicker.bjl

AnotherDatePicker.bas 是跨平台自定义视图文件。

Manifest.txt 包含：

Version=2.00	版本号。
B4J.DependsOn=jXUI, jDateUtils	用于 B4J 的库。
B4A.DependsOn=XUI, DateUtils	用于 B4A 的库。
B4i.DependsOn=iXUI, iDateUtils	用于 B4i 的库。

将 xxx.b4xlib 文件复制到 AdditionalLibraries\B4X 文件夹。

如果有 xxx.xml 文件，则不能将其保存在那里，而应保存在另一个文件夹中。

B4XLibraries 在 [B4X 自定义视图手册](#) 中有说明。

### 4.9.4 加载和更新库

在 B4X 网站上可以找到官方和附加库的列表以及相关帮助文档的链接：

B4A Documentation page: [List of Libraries.](#)

B4i Documentation page: [List of Libraries.](#)


B4J Documentation page: [List of Libraries.](#)

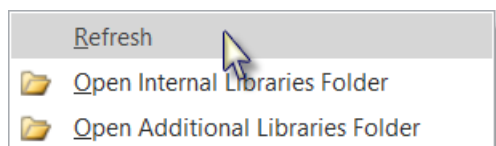
B4R Documentation page: [List of Libraries.](#)

Or in the [B4X Libraries Google sheet.](#)

要查找库文件，请在您的互联网浏览器中使用类似  
<http://www.b4x.com/search?query=betterdialogs+library>  
的查询。

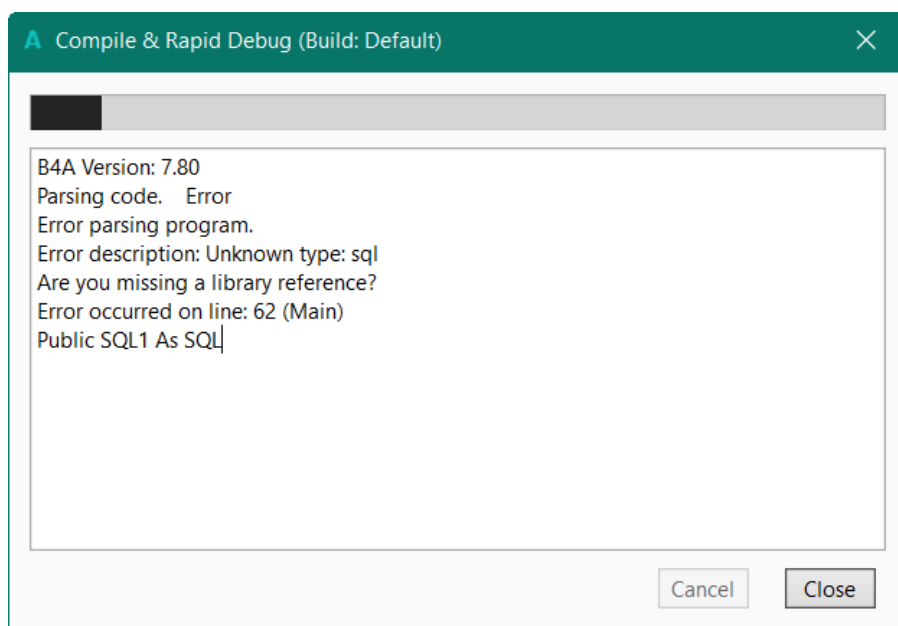
要加载或更新库，请按照以下步骤操作：

- 将库 zip 文件下载到某处。
- 把它解压。
- 将 xxx.jar 和 xxx.xml 文件复制到
  - B4X 库文件夹（用于标准 B4X 库）
  - [附加库文件夹](#) 附加库文件夹（用于附加库）。
- 右键单击 [库管理器选项卡](#)，然后单击  并选择库。



### 4.9.5 错误消息“您是否缺少库引用？”

如果您收到类似这样的消息，则意味着您忘记检查 Lib Tab 列表中的指定库！



## 4.9.6 在哪里可以找到库？

要查找库，您可以：

- 在论坛中搜索其名称。
- 或查看在线库索引。

### 4.9.6.1 在线库索引

您可以通过以下链接查看在线库索引：

[https://docs.google.com/spreadsheets/d/1qFvc3Q70RriJS3m\\_ywBoJvZ47gSTVAuN\\_X04SiO\\_XBw/edit#gid=0](https://docs.google.com/spreadsheets/d/1qFvc3Q70RriJS3m_ywBoJvZ47gSTVAuN_X04SiO_XBw/edit#gid=0)

截屏：

1	Library Name	Short Description	Files Names (without extension)				Last Update	
			B4A	B4i	B4J	B4R	Version	Date
3	MFRC522	RFID reader / writer				rMFRC522	1.02	15-Mar-2019
4	4 Button B4xDialog	Additional button for B4xDialog	B4xDialog4Button	B4xDialog4Button	B4xDialog4Button		1.1	21-Dec-2020
5	433MHz R/T	Support for 433MHz receiver and transmitter				rRCSwitch	1.01	5-Aug-2018
6	ABMaterial	WebApps Framework with Materialize CSS			ABMaterial		4.51	18-Nov-2018
7	ABMServer	WebApps Mini Template for ABMaterial			ABMServer		1.07	28-Feb-2021
8	ActivityRecognition	Monitor the user state	ActivityRecognition (class)				3	27-May-2020
9	AdColonyAds	AdColony Ads	AdColonyAds				4.65	13-Jan-2022
10	AdManager	AdManager Ads	AdManager				1.52	22-Feb-2021
11	Administrator	Android administrator features	Administrator				1.1	11-Sep-2017
12	AmazonAds	Amazon Ads	AmazonAds				1	24-Sep-2020

Last Update		Author	IDE Comment	Forum Link
Version	Date			
1.02	15-Mar-2019	Erel		<a href="https://www.b4x.com/android/forum/threads/mfrc522-rfid-reader-writer.67">https://www.b4x.com/android/forum/threads/mfrc522-rfid-reader-writer.67</a>
1.1	21-Dec-2020	Steve105		<a href="https://www.b4x.com/android/forum/threads/b4x-b4xdialog4button.1241/">https://www.b4x.com/android/forum/threads/b4x-b4xdialog4button.1241/</a>
1.01	5-Aug-2018	JanDerKan		<a href="https://www.b4x.com/android/forum/threads/rcswitch-library.95830/">https://www.b4x.com/android/forum/threads/rcswitch-library.95830/</a>
4.51	18-Nov-2018	Alwaysbusy		<a href="https://www.b4x.com/android/forum/threads/abmaterial-framework-for-w">https://www.b4x.com/android/forum/threads/abmaterial-framework-for-w</a>
1.07	28-Feb-2021	Alwaysbusy		<a href="https://www.b4x.com/android/forum/threads/abmaterial-abmsener-mini-1">https://www.b4x.com/android/forum/threads/abmaterial-abmsener-mini-1</a>
3	27-May-2020	Erel		<a href="https://www.b4x.com/android/forum/threads/physical-activity-recognition">https://www.b4x.com/android/forum/threads/physical-activity-recognition</a>
4.65	13-Jan-2022	Pendrush		<a href="https://www.b4x.com/android/forum/threads/adcolony-library.120665/">https://www.b4x.com/android/forum/threads/adcolony-library.120665/</a>
1.52	22-Feb-2021	Pendrush		<a href="https://www.b4x.com/android/forum/threads/admanager-library.122111/">https://www.b4x.com/android/forum/threads/admanager-library.122111/</a>
1.1	11-Sep-2017	Erel		<a href="https://www.b4x.com/android/forum/threads/device-administrator-library">https://www.b4x.com/android/forum/threads/device-administrator-library</a>
1	24-Sep-2020	Pendrush		<a href="https://www.b4x.com/android/forum/threads/amazonads-library.122691/">https://www.b4x.com/android/forum/threads/amazonads-library.122691/</a>

您将找到：

- 库名称。
- 简短描述。
- 文件名（不带扩展名）和相关平台。
- 上次更新：最新版本和更新日期。
- 作者
- IDE 评论此评论将显示在库管理器中的 IDE 中。
- 论坛链接：此链接将引导您进入找到库的论坛主题。

## 4.10 字符串操作

### 4.10.1 B4A, B4i, B4J 字符串

B4A, B4i 和 B4J 允许像其他 Basic 语言一样进行字符串操作，但有一些区别。

这些操作可以直接在字符串上完成。

示例：

```
txt = "123,234,45,23"
txt = txt.Replace(",", ";")
```

结果：123;234;45;23

不同的功能包括：

- **CharAt (Index)** 返回给定索引处的字符。
- **CompareTo (Other)** 按字典顺序将该字符串与其他字符串进行比较。
- **Contains (SearchFor)** 测试字符串是否包含给定的 SearchFor 字符串。
- **EndsWith (Suffix)** 如果字符串以给定的后缀子字符串结尾，则返回 True。
- **EqualsIgnoreCase (Other)** 如果两个字符串相等（忽略大小写），则返回 True。
- **GetBytes (Charset)** 将 Charset 字符串编码为新的字节数组。
- **IndexOf (SearchFor)** 返回字符串中 SearchFor 第一次出现的索引。索引从 0 开始。如果未找到任何结果，则返回 -1。
- **IndexOf2 (SearchFor, Index)** 返回字符串中 SearchFor 第一次出现的索引。从给定的索引开始搜索。索引从 0 开始。如果未找到任何结果，则返回 -1。
- **LastIndexOf (SearchFor)** 返回字符串中 SearchFor 第一次出现的索引。搜索从字符串的末尾开始，然后前进到开头。索引从 0 开始。如果未找到任何结果，则返回 -1。
- **LastIndexOf2 (SearchFor)** 返回字符串中 SearchFor 第一次出现的索引。搜索从给定的索引开始并前进到开头。索引从 0 开始。如果未找到任何结果，则返回 -1。
- **Length** 返回字符串的长度，字符数。
- **Replace (Target, Replacement)** 返回用 Replacement 替换所有出现的 Target 后产生的新字符串。
- **StartsWith (Prefix)** 如果此字符串以给定的前缀开头，则返回 True。
- **Substring (BeginIndex)** 返回一个新字符串，它是原始字符串的子字符串。新的字符串将包含 BeginIndex 处的字符并延伸到字符串的末尾。
- **Substring2 (BeginIndex, EndIndex)** 返回一个新字符串，它是原始字符串的子字符串。新字符串将包含 BeginIndex 处的字符并延伸到 EndIndex 处的字符，但不包括最后一个字符。  
请注意，EndIndex 是结束索引，而不是其他语言中的长度。
- **ToLowerCase** 返回将该字符串小写化后的新字符串。
- **ToUpperCase** 返回将该字符串大写后的新字符串。
- **Trim** 返回原始字符串的副本，不包含任何前导或尾随空格。

**注意：**字符串函数区分大小写。

如果您想使用不区分大小写的函数，则应使用 ToLowerCase 或 ToUpperCase。

例子：NewString = OriginalString.ToLowerCase.StartsWith("pre")

### 4.10.2 字符串连接

连接字符串的连接字符是：&

例子：

- 字符串  
`Private MyString As String`  
`MyString = "aaa" & "bbb" & "ccc"`      结果： aaabbbccc
- 字符串和数字  
`MyString = "$: " & 1.25`      结果： \$: 1.25
- 字符串和变量，它可以是另一个字符串或数字。  
`Private Val As Double`  
`Val = 1.25`  
`MyString = "$: " & Val`      结果： \$: 1.25

不要与 VB 语法混淆：

```
MyString = "aaa" + "bbb" + "ccc"
```

这不管用！



### 4.10.3 B4A, B4i, B4J 字符串生成器

StringBuilder 是一个可变字符串，与不可变的常规字符串不同。  
当您需要连接多个字符串时，StringBuilder 特别有用。

下面的代码演示了 StringBuilder 的性能提升：

```
Dim start As Long
start = DateTime.Now
'常规字符串
Dim s As String
For i = 1 To 5000
    s = s & i
Next
Log(DateTime.Now - start)
'字符串生成器
start = DateTime.Now
Dim sb As StringBuilder
sb.Initialize
For i = 1 To 5000
    sb.Append(i)
Next
Log(DateTime.Now - start)
```

在真实设备上测试，第一个“for 循环”大约需要 20 秒，第二个需要不到十分之一秒。  
原因是代码：s = s & i 每次迭代都会创建一个新字符串（字符串是不可变的）。  
方法 StringBuilder.ToString 将对象转换为字符串。

### 4.10.3.1 StringBuilder 方法

**Append** (Text As String) As StringBuilder

在末尾附加指定文本。

返回相同的对象，因此您可以链接方法。

示例：

```
sb.Append("First line").Append(CRLF).Append("Second line")
```

**Initialize**

初始化对象。

例子：

```
Dim sb As StringBuilder
```

```
sb.Initialize
```

```
sb.Append("The value is: ").Append(其他变量).Append(CRLF)
```

**Insert** (Offset As Int, Text As String) As StringBuilder

在指定偏移量处插入指定文本。

**IsInitialized** As Boolean

是否已初始化为布尔值

**Length** As Int [只读]

返回字符数。

**Remove** (StartOffset As Int, EndOffset As Int) As StringBuilder

删除指定的字符。

StartOffset - 要删除的第一个字符。

EndOffset - 结束索引。此字符不会被删除。

**ToString** As String

将对象转换为字符串。

### 4.10.4 智能字符串文字

“智能字符串”文字是标准字符串文字的更强大版本。

它具有三个优点：

1. 支持多行字符串。
2. 无需转义引号。
3. 支持字符串插值。

智能字符串文字以 `$` 开头，以 `"$"` 结尾。

例子：

```
Dim s As String = $"Hello world"$
Dim query As String = $
SELECT value_id FROM table3
WHERE rowid >= random()%(SELECT max(rowid)FROM table3)
AND second_value ISNOTNULL
LIMIT 1"$
Log($"无需转义“引号”！ "$)
```

#### 4.10.4.1 字符串插值

智能字符串可以保存零个或多个带有代码的占位符。占位符可以轻松格式化。

占位符以 `$[可选格式化程序]{` 开头，以 `}` 结尾：

```
Log($"5 * 3 = ${5 * 3}"$) '5 * 3 = 15
```

您可以在占位符内放置任何您喜欢的代码。

```
Dim x = 1, y = 2, z = 4 As Int
Log($"x = ${x}, y = ${y}, z = ${Sin(z)}"$) 'x = 1, y = 2, z = -0.7568024953079282
```

这是一个编译时功能。例如，您无法从文件加载字符串。

#### 4.10.4.2 数字格式化程序

数字格式化程序允许您设置整数的最小数量和小数的最大数量。它类似于 `NumberFormat` 关键字。

数字格式化程序结构：`MinIntegers.MaxFractions`。`MaxFractions` 组件是可选的。

示例：

```
Dim h = 2, m = 15, s = 7 As Int
Log($"剩余时间 $2{h}:$2{m}:$2{s}"$) '剩余时间 02:15:07
Log($"10 / 7 = $0.3{10 / 7}"$) '10 / 7 = 1.429
Log($"$1.2{"该值不是数字!"}"$) 'NaN
```

#### 4.10.4.3 其他格式化程序

请注意，格式化程序不区分大小写。

**Date** - 相当于 `DateTime.Date`:

```
Log($"当前日期为 $date{DateTime.Now}") '当前日期为 02/02/2015
```

**Time** - 相当于 `DateTime.Time`:

```
Log($"当前时间为 $time{DateTime.Now}") '当前时间为 11:17:45
```

**DateTime** - 相当于 `DateTime.Date & " " & DateTime.Time`:

```
Log($"当前时间为 $DateTime{DateTime.Now}") '当前时间为 02/02/2015 11:18:36
```

**XML** - 转义五个 XML 实体 (", ', <, >, &):

```
Dim UserString As String = $"它会破坏您的解析器 ><'&?"$  
Log($"用户输入是: $xml{UserString}")  
'用户输入是: 它会破坏您的解析器 &gt;&lt;&#39;&quot;&amp;?
```

这对于 html 内容也很有用。

### 4.10.5 B4A, B4i 字符序列 CS 生成器

CharSequence 是 Android SDK 中的原生接口。

String 是 CharSequence 的一种实现。

CharSequence 还有其他实现，它们提供了更多功能，允许我们格式化字符串，添加图像，甚至使文本的某些部分可点击。

从 B4A v6.80 开始，许多方法都接受 CharSequence 而不是 String。现有代码将正常工作，因为您可以传递常规字符串。但是，您现在还可以传递更有趣的 CharSequence。

**库开发人员请注意**，如果您的库调用与 CharSequence 一起使用的 API，那么您应该将方法签名更改为使用 CharSequence 而不是 String。这将允许开发人员格式化文本。

本教程介绍 CSBuilder 对象。

CSBuilder 与 StringBuilder 类似。它不是构建字符串，而是构建包含样式信息的 CharSequence。

这些示例是用 B4A 制作的，但 B4i 的原理相同

使用它非常简单。

#### 4.10.5.1 文本

```
Private cs As CSBuilder
cs = cs.Initialize.Color(Colors.Red).Append("Hello World!").PopAll
Label1.Text = cs
```



默认背景颜色可能因 Android 版本的不同而不同。

CSBuilder 的几乎所有方法都会返回对象本身。这使我们能够链接方法调用。

文本始终使用 Append 方法附加。

可以设置各种属性。设置属性标记样式跨度的开始。

调用 Pop 会结束最后添加的跨度（尚未结束）。

调用 PopAll 会结束所有打开的跨度。始终在末尾调用 PopAll 以确保所有跨度都已关闭，这很方便。

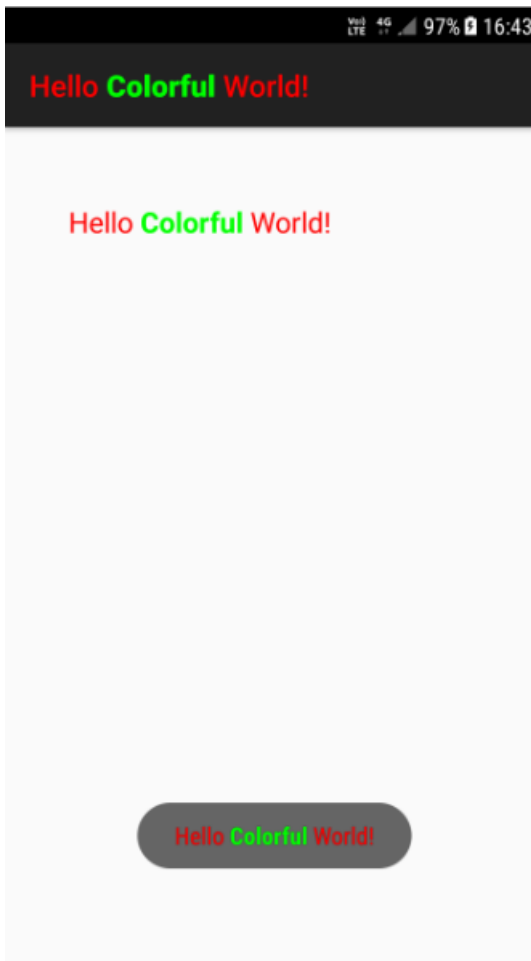
**明确弹出属性的示例：**

```
Label1.Text = cs.Initialize.Color(Colors.Red).Append("Hello  
").Pop.Append("World!").PopAll
```



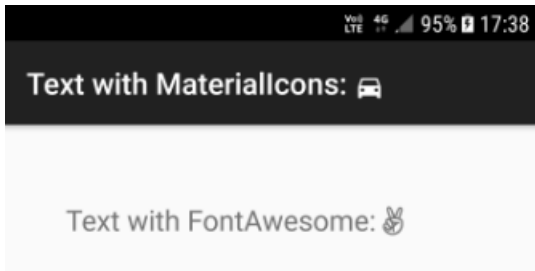
这些方法是链接在一起的还是分成几行并不重要：

```
Private cs As CSBuilder
cs.Initialize.Color(Colors.Red).Append("Hello ")
cs.Bold.Color(Colors.Green).Append("Colorful ").Pop.Pop
'两次弹出：第一次移除绿色，第二次移除粗体样式
cs.Append("World!").PopAll
Label1.Text = cs
'也可以设置为活动标题
Activity.Title = cs
'和 Toast 消息以及其他地方...
ToastMessageShow(cs, True)
```



### 4.10.5.2 使用 FontAwesome 或 MaterialIcons

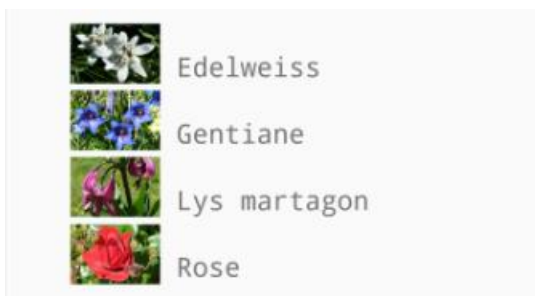
```
Private cs As CSBuilder
Label1.Text = cs.Initialize.Append("Text with FontAwesome:
").Typeface(Typeface.FONTAWESOME).Append(Chr(0xF209)).PopAll
'多次使用同一个构建器。请注意，每次都会初始化它。
'请注意，我们垂直对齐了材质图标字符。
cs.Initialize.Append("Text with MaterialIcons:
").Typeface(Typeface.MATERIALICONS).VerticalAlign(5dip).Append(Chr(0xE531)).PopAll
Activity.Title = cs
```



**注意：**MaterialIcons 字符的十六进制值以 0xE 开头，FontAwesome 字符的十六进制值以 0xF 开头

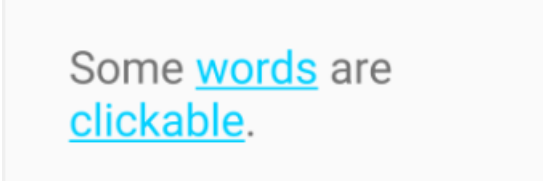
### 4.10.5.3 图片

```
Private cs As CSBuilder
cs.Initialize.Size(18).Typeface(Typeface.MONOSPACE)
cs.Image(LoadBitmap(File.DirAssets, "edelweiss.jpg"), 60dip, 40dip, False).Append("
Edelweiss").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "gentiane.jpg"), 60dip, 40dip, False).Append("
Gentiane").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "lys_martagon.jpg"), 60dip, 40dip, False).Append("
Lys martagon").Append(CRLF)
cs.Image(LoadBitmap(File.DirAssets, "rose.jpg"), 60dip, 40dip, False).Append("
Rose").Append(CRLF)
cs.PopAll
Label1.Text = cs
```



#### 4.10.5.4 可点击文本

Clickable 方法创建可点击的文本。要引发该事件，您必须调用 `cs.EnableClickEvents`。  
`Append` 方法接受 `CharSequence`。在下面的代码中，`CreateClickableWord` 子程序返回一个 `CharSequence`，然后将其附加到另一个 `CharSequence`。



#### 4.10.5.5 突出显示文本

来自 [SearchView](#) 类的示例。

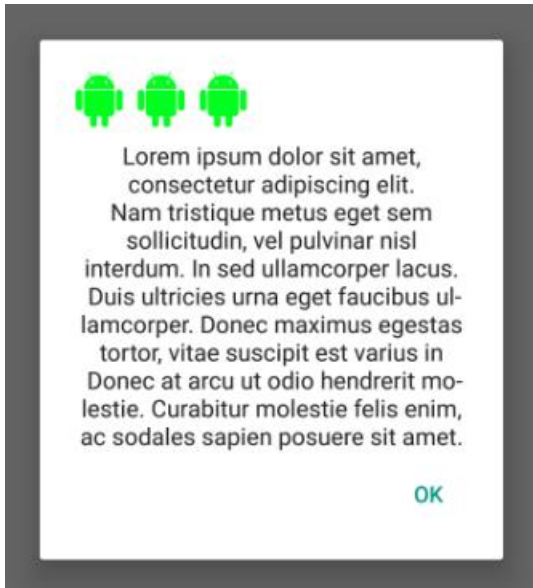
```
Private Sub AddItemsToList(li As List, full As String)
    If li.IsInitialized = False Then Return
    Dim cs As CSBuilder
    For i = 0 To li.Size - 1
        Dim item As String = li.Get(i)
        Dim x As Int = item.ToLowerCase.IndexOf(full)
        If x = -1 Then
            Continue
        End If
        cs.Initialize.Append(item.SubString2(0,
x)).Color(highlightColor).Append(item.SubString2(x, x + full.Length)).Pop
        cs.Append(item.SubString(x + full.Length))
        lv.AddSingleLine(cs)
    Next
End Sub
```





#### 4.10.5.6 居中对齐文本

```
Msgbox(cs.Initialize.Alignment("ALIGN_CENTER").Append($"Lorem ipsum dolor sit am  
et, consectetur adipiscing elit.  
Nam tristique metus eget sem sollicitudin, vel pulvinar nisl interdum. In sed ul  
lamcorper lacus.  
Duis ultricies urna eget faucibus ullamcorper. Donec maximus egestas tortor, vit  
ae suscipit est varius in  
Donec at arcu ut odio hendrerit molestie. Curabitur molestie felis enim, ac soda  
les sapien posuere sit amet."$).PopAll, _  
cs.Initialize.Typeface(Typeface.FONTAWESOME).Color(0xFF01FF20).Size(40).Append(C  
hr(0xF17B) & " " & Chr(0xF17B) & " " & Chr(0xF17B)).PopAll)
```



### 4.10.5.7 CSBuilder 方法

#### 4.10.5.7.1 B4A / B4i

- **Alignment** (Alignment As Alignment Enum)  
开始对齐跨度。  
对齐 - 以下字符串之一：  
ALIGN\_NORMAL, ALIGN\_OPPOSITE 或 ALIGN\_CENTER
- **Append** (Text As CharSequence)  
附加提供的字符串或字符序列。
- **BackgroundColor** (Color As Int)  
开始背景颜色跨度。
- **Color** (Color As Int)  
开始前景色跨度。
- **Initialize**  
初始化构建器。您可以多次调用此方法来创建新的 CharSequence。  
请注意，与大多数其他方法一样，它返回当前对象。
- **IsInitialized**  
测试此对象是否已初始化。返回布尔值。
- **Pop**  
关闭最近的跨度。所有跨度都必须关闭。您可以调用 PopAll 来关闭所有打开的跨度。
- **PopAll**  
关闭所有打开的跨度。  
最后总是调用 PopAll 来确保所有跨度都已关闭，这样很方便。
- **Strikethrough**  
开始删除线范围。
- **ToString**  
返回带有字符的字符串。
- **Underline**  
开始下划线跨越。
- **VerticalAlign** (Shift As Int)  
开始垂直对齐跨度（正数 = 向下）。

**4.10.5.7.2 仅限 B4A**

- **Bold**  
开始一个粗体跨度。
- **Clickable** (EventName As String, Tag As Object)  
开始一个可点击的跨度。对于要引发的事件，您需要调用 `EnableClickEvents` 方法。

例子：

```
Sub Activity_Create(FirstTime As Boolean)
    Activity.LoadLayout("1")
    Dim cs As CSBuilder
    cs.Initialize.Size(30).Append("Some ").Append(CreateClickableWord("words"))
    cs.Append(" are ").Append(CreateClickableWord("clickable")).Append(".").PopAll
    Label1.Text = cs
    cs.EnableClickEvents(Label1)
End Sub

Sub CreateClickableWord(Text As String) As CSBuilder
    Dim cs As CSBuilder
    Return cs.Initialize.Underline.Color(0xFF00D0FF).Clickable("word", Text).Append(Text).PopAll
End Sub

Sub Word_Click (Tag As Object)
    Log($"您点击了单词: ${Tag}")
End Sub
```

- **EnableClickEvents** (Label As TextView)  
使用可点击跨度时应调用此方法。
- **Image** (Bitmap As Bitmap, Width As Int, Height As Int, Baseline As Boolean)  
添加图像跨度。此方法将添加一个空格字符作为图像的占位符。  
与其他方法不同，您不需要调用 `Pop` 来关闭此跨度，因为它会自动关闭。  
Bitmap 位图 - 图像。  
Width 宽度 / Height 高度 - 图像尺寸，使用 ‘dip’ 单位。  
Baseline - 如果为 `true`，则图像将根据基线对齐。  
否则，它将根据文本中的最低下标对齐。
- **RelativeSize** (Proportion As Float)  
开始一个相对大小的跨度。实际文本大小将乘以设置的 `Proportion`。
- **ScaleX** (Proportion As Float)  
开始一个比例 X 跨度。它水平缩放文本。
- **Size** (Size As Int)  
开始一个文本大小范围。请注意，您不应使用带有文本大小尺寸的 ‘dip’ 单位。
- **TypeFace** (Typeface As Typeface)  
启动自定义字体跨度。  
类似于 B4i 的字体。

**4.10.5.7.3 B4i only**

- **Font** (Font As B4IFontWrapper)  
开始字体跨度。  
请注意，调用 `AutoScaleAll` 时字体会重置。  
您应该在父 `Resize` 事件中更改字体，或从布局设计器脚本中删除对 `AutoScaleAll` 的调用。  
类似于 B4A 的 `TypeFace`。
- **KerningScale** (Scale As Float)  
设置字距（水平间距）比例。
- **Link** (URL As NSString)  
创建链接。链接将在不可编辑的 `TextView` 中可点击。

### 4.10.6 B4J TextFlow 类

[TextFlow 类](#) 使用 `JsonObject` 创建 `TextFlow` 节点。使用 `TextFlow`，您可以显示具有不同颜色，字体和其他属性的富文本。

用法：

- 将 `TextFlow` 类模块添加到您的项目（工具 - 添加现有模块）。
- 创建 `TextFlow` 对象。
- 调用 `AddText` 添加文本部分并设置其属性。
- 最终您应该调用 `CreateTextFlow` 来创建将添加到布局的节点。

请注意，设置的属性返回允许链接调用的类实例。

示例代码：

```
Dim tf As TextFlow
tf.Initialize
tf.AddText("1 2 3").SetColor(fx.Colors.Red).SetUnderline(True)
tf.AddText(" 4 5 6 ").SetColor(fx.Colors.Green).SetFont(fx.CreateFont("", 17, True, True))
tf.AddText("7 8 9").SetColor(fx.Colors.Blue).SetStrikethrough(True).SetFont(fx.DefaultFont(20))
Dim pane As Pane = tf.CreateTextFlow
MainForm.RootPane.AddNode(pane, 10, 10, 200, 100)
```

### 4.10.7 B4R

B4R 不支持其他 Basic 语言中的字符串操作。

此类操作可以通过 rRandomAccesFile 库中的 ByteConverter 对象完成。

B4R 字符串与其他 B4X 工具中的字符串不同。造成这些差异的原因是：

- 内存非常有限。
- 缺少 Unicode 编码器。

B4R 中的 String 对象与 C 语言 char\* 字符串相同。它是一个字节数组，末尾有一个额外的零字节。

最后一个零字节的要求使得无法在不将内存复制到新地址的情况下创建子字符串。

**因此，字节数组比字符串更可取。**

各种与字符串相关的方法都适用于字节数组。

将字符串转换为字节数组非常简单，不涉及任何内存复制。编译器会在需要时自动执行此操作：

```
Private b() As Byte = "abc" '相当于 Private b() As Byte = "abc".GetBytes
```

仅支持两个功能：

这些功能包括：

- |                            |                                      |
|----------------------------|--------------------------------------|
| • <b>GetBytes(Charset)</b> | 将字符串内容作为字节数组返回。<br>请注意，数组和字符串共享相同的内存 |
| • <b>Length</b>            | 返回字符串的长度，字符数。                        |

## 字符串方法

标准字符串方法在 ByteConverter 类型（rRandomAccessFile 库）中可用。

它们与其他 B4X 工具中的字符串方法类似：

```
Private Sub AppStart
    Serial1.Initialize(115200)
    Log("AppStart")
    Dim bc As ByteConverter
    Log("索引: ", bc.IndexOf("0123456", "3")) '索引: 3
    Dim b() As Byte = " abc,def,ghijkl "
    Log("子字符串: ", bc.SubString(b, 3)) '子字符串: c,def,ghijkl
    Log("修剪: ", bc.Trim(b)) '修剪: abc,def,ghijkl
    For Each s() As Byte In bc.Split(b, ",")
        Log("分裂: ", s)
        '分裂: abc
        '分裂: def
        '分裂: ghijkl
    Next
    Dim c As String = JoinStrings(Array As String("毫秒数: ", Millis, CRLF, "微机数
量: ", Micros))
    Log("c = ", c)
    Dim b() As Byte = bc.SubString2(c, 0, 5)
    b(0) = Asc("X")
    Log("b = ", b)
    Log("c = ", c) '第一个字符将是 X
End Sub
```

请注意，字符串和字节数组都可以使用，因为编译器会自动将字符串转换为字节数组。

除 JoinStrings 外，上述方法均不会复制原始字符串/字节。

这意味着修改返回的数组（如最后三行所示）也会修改原始数组。

这也会发生在共享相同内存块的字符串文字上：

```
Private Sub AppStart
    Serial1.Initialize(115200)
    Log("AppStart")
    Dim bc As ByteConverter
    Dim b() As Byte = bc.Trim("abcdef ")
    b(0) = Asc("M") '此行将改变文字字符串的值
    Dim s As String = "abcdef "
    Log(s) 'Mbcdef
End Sub
```

rRandomAccessFile 库中的 ByteConverter 对象中的字符串操作：

- **EndsWith(Source As Byte(), Suffix As Byte())**  
如果字符串以给定的后缀子字符串结尾，则返回 True。
- **IndexOf(Source As Byte(), SearchFor As Byte())**  
返回字符串中第一次出现的 SearchFor 的索引。
- **IndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**  
返回字符串中 SearchFor 第一次出现的索引。从给定的索引开始搜索。
- **LastIndexOf(Source As Byte(), SearchFor As Byte())**  
返回源字符串中 SearchFor 第一次出现的索引。  
从字符串末尾开始搜索。
- **LastIndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**  
返回源字符串中 SearchFor 第一次出现的索引。  
从给定的索引开始搜索并前进到开头。
- **StartsWith(Source As Byte(), Prefix As Byte())**  
如果此字符串以给定的前缀开头，则返回 True。
- **Substring(Source As Byte(), BeginIndex As UInt)**  
返回一个新字符串，它是原始字符串的子字符串。  
新字符串将包含 BeginIndex 处的字符并延伸到字符串的末尾。
- **Substring2(Source As Byte(), BeginIndex As UInt, EndIndex As UInt)**  
返回一个新字符串，它是原始字符串的子字符串。新字符串将包含 BeginIndex 处的字符并延伸到 EndIndex 处的字符，但不包括最后一个字符。
- **Trim(Source As Byte())**  
返回原始字符串的副本，不包含任何前导或尾随空格。



## 4.11 数字格式

### 4.11.1 B4A, B4i, B4J

数字格式化，将数字显示为不同格式的字符串，有两个关键字：

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)  
 NumberFormat(12345.6789, 0, 2) = 12,345.68  
 NumberFormat(1, 3, 0) = 001  
 NumberFormat(Value, 3, 0) 可以使用变量。  
 NumberFormat(Value + 10, 3, 0) 可以使用算术运算。  
 NumberFormat((lblscore.Text + 10), 0, 0) 如果一个变量是字符串，则添加括号。
- **NumberFormat2**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean)  
 NumberFormat2(12345.67, 0, 3, 3, True) = 12,345.670  
 NumberFormat2(12345.67, 0, 3, 3, False) = 12345.670

### 4.11.2 B4X 数字格式化程序

[B4XFormatter](#) 是 NumberFormat / NumberFormat2 关键字的替代方案。它在 B4X 中作为 b4xlib 实现，并且是跨平台的。

库中有两种类型：

B4XFormatter - 主类。

B4XFormatData - 具有各种可配置字段的类型。

格式化程序包含格式数据对象列表。新的格式化程序以单一格式数据作为默认格式开始。

### 4.11.3 B4R

数字格式化，将数字显示为具有不同格式的字符串：

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)  
 NumberFormat(12345.6789, 0, 2) = 12,345.68  
 NumberFormat(1, 3, 0) = 001  
 NumberFormat(Value, 3, 0) 可以使用变量。  
 NumberFormat(Value + 10, 3, 0) 可以使用算术运算。  
 NumberFormat((lblscore.Text + 10), 0, 0) 如果一个变量是字符串，则添加括号。

## 4.12 计时器

计时器 `Timer` 对象以指定的间隔生成 `Tick` 事件。使用计时器是长循环的一个很好的替代方案，因为它允许 UI 线程处理其他事件和消息。

请注意，当 UI 线程忙于运行其他代码时，计时器事件不会触发。

当活动暂停时，或者当阻塞对话框（如 `Msgbox`）可见时，计时器事件不会触发。

在 B4A 中，在活动暂停时禁用计时器，然后在活动恢复时启用它也很重要。这将节省 CPU 和电池。

计时器具有：

- 三个参数。
  - **Initialize** 使用两个参数初始化计时器，即 `EventName` 和间隔。  
`Timer1.Initialize(EventName As String, Interval As Long)`  
例如: `Timer1.Initialize("Timer1", 1000)`
  - **Interval** 设置计时器间隔（以毫秒为单位）。  
`Timer1.Interval = Interval`  
例如: `Timer1.Interval = 1000`, 1 秒
  - **Enabled** 启用或禁用计时器。默认情况下为 `False`。  
例如: `Timer1.Enabled = True`
- 一个事件
  - **Tick** 每个时间间隔都会调用 `Tick` 例程。  
例如: `Sub Timer1_Tick`

必须在 `Process_Global` 例程中声明计时器。

```
Sub Process_Globals
    Public Timer1 As Timer
```

但是必须在使用定时器滴答事件例程的模块中的下列例程之一中对其进行初始化。

B4A: Activity\_Create 常规

```
Sub Activity_Create(FirstTime As Boolean)
    If FirstTime = True Then
        Timer1.Initialize("Timer1", 1000)
    End If
```

B4i: Application\_Start 常规

```
Private Sub Application_Start (Nav As NavigationController)
    Timer1.Initialize("Timer1", 1000)
```

B4J: AppStart 常规

```
Sub AppStart (Form1 As Form, Args() As String)
    Timer1.Initialize("Timer1_Tick", 1000)
```

B4R: AppStart 常规

```
Private Sub AppStart
    Timer1.Initialize("Timer1", 1000)
```

以及 Timer Tick 事件例程。

操作系统每秒（1000 毫秒）都会调用此例程。

```
Private Sub Timer1_Tick
    '做点什么
End Sub
```

## 4.13 文件 B4A, B4i, B4J

许多应用程序需要访问持久存储。最常见的两种存储类型是文件和数据库。

Android 和 iOS 有自己的文件系统。B4A 和 B4i 程序都无法访问 Windows 系统中的文件。

要将文件添加到您的项目中，您必须在 IDE 中的“文件”选项卡中添加这些文件。这些文件将添加到项目“文件”文件夹中。

### 4.13.1 文件对象

预定义对象 `File` 具有许多用于处理文件的函数。

#### 4.13.1.1 文件位置

有几个重要的位置可以读取或写入文件。

##### `File.DirAssets`

`assets` 文件夹包含使用 IDE 中的文件管理器添加的文件。

它是项目文件夹中的 `Files` 文件夹。

##### **这些文件是只读的！**

您无法在此文件夹中创建新文件（实际上位于 `apk` 文件中）。

如果 `Dir.Assets` 文件夹中有数据库文件，则需要将其复制到另一个文件夹才能使用它。

##### 4.13.1.1.1 B4X

为了保存由应用程序生成且仅由应用程序使用的数据，您可以使用 `xui`（`jxui` 或 `ixui`）库获取默认文件夹。

##### `xui.DefaultFolder`

此文件夹与以下文件夹相同：

- B4A - 与 `File.DirInternal` 相同。
- B4i - 与 `File.DirDocuments` 相同。
- B4J - 与 `File.DirData` 相同。

您必须先调用一次 `SetDataFolder`，然后才能使用此文件夹。

**`xui.SetDataFolder(AppName As String)`**

#### 4.13.1.1.2 仅限 B4A

##### **File.DirInternal / File.DirInternalCache**

这两个文件夹存储在设备的主内存中，仅供您的应用程序使用。其他应用程序无法访问这些文件。如果需要更多空间，操作系统可能会删除缓存文件夹。

##### **File.DirRootExternal 仅在真正需要时才使用此文件夹。**

存储卡根文件夹。大多数情况下，这是内部存储卡，而不是外部 SD 卡。

##### **File.DirDefaultExternal**

SD 卡中应用程序的默认文件夹。

该文件夹为：<存储卡>/Android/data/<包>/files/

如果需要，将创建该文件夹。

请注意，调用上述两个属性中的任何一个都会为您的应用程序添加 `EXTERNAL_STORAGE` 权限。

提示：您可以使用 `File.ExternalReadable` 和 `File.ExternalWritable` 检查是否存在存储卡以及是否可用。

##### **外部存储。**

您应该使用 `RuntimePermissions` 库来获取最佳文件夹：

```
MyFolder = RuntimePermissions.GetSafeDirDefaultExternal(SubFolder As String)
```

返回应用程序在辅助存储设备上的默认文件夹的路径。

如果没有可用的辅助存储，则将返回 `File.DirInternal` 的路径。

它是 `File.DirDefaultExternal` 的更好替代方案。

在 Android 4.4+ 上，无需任何权限即可访问此文件夹。

`SubFolder` - 将为您的应用程序创建的子文件夹。如果不需要，请传递一个空字符串。

在 Android 中，访问外部存储设备中的文件变得很麻烦。

Erel 编写了一个类 `ExternalStorage` - 访问 SD 卡和 USB 棒以“简化”访问。

摘自 Erel 的线程：

在我们开始之前：

1. 外部存储是指真正的 SD 卡或连接的大容量存储 USB 设备。
2. 它与 `File.DirRootExternal / DirDefaultExternal` 无关，后者实际上指向内部存储。
3. 它与运行时权限无关。
4. 您可以使用 `RuntimePermissions.GetAllSafeDirsExternal` 直接访问 SD 卡上的特定文件夹。
5. 此类的最低版本为 Android 5。它可能适用于 Android 4.4（如果您想尝试，请更改 `minSdkVersion`）。

从 Android 4.4 开始，不再可能直接访问外部存储。

访问这些存储的唯一方法是通过存储访问框架（SAF），这是一个相当复杂且文档不足的框架。

ExternalStorage 类使使用 SAF 变得更加简单。

用法：

1. 调用 ExternalStorage.SelectDir。这将打开一个对话框，允许用户选择根文件夹。一旦选择，根文件夹的 uri 就会被存储，以后可以使用，而无需用户再次选择文件夹。即使在设备启动后也是如此。

2. 等待 ExternalFolderAvailable 事件。

现在您可以访问 Storage.Root 下的文件，包括子文件夹内的文件。

3. 文件表示为名为 ExternalFile 的自定义类型。

4. 支持以下操作：ListFiles, Delete, CreateNewFile, FindFile, OpenInputStream 和 OpenOutputStream。

参见附件示例。

依赖于：ContentResolver 和 JavaObject 库。

添加：

#AdditionalJar: com.android.support:support-core-utils

#### 4.13.1.1.3 仅限 B4i

##### **File.DirDocuments**

文档文件夹仅用于存储用户生成的内容。可以通过 iTunes 共享此文件夹。  
此文件夹由 iTunes 自动备份。

##### **File.DirLibrary**

任何非用户生成的持久文件的位置。此文件夹由 iTunes 自动备份。  
您可以创建一个名为 Caches 的子文件夹。该文件夹下的文件将不会被备份。

##### **File.DirTemp**

临时文件夹。此文件夹中的文件不会由 iTunes 备份，可能会不时被删除。

#### **B4i 访问外部资源或共享到外部应用程序的方法。**

论坛中的这个主题显示了一些共享文件的方法：  
[访问外部资源或共享到外部应用程序的方法列表。](#)

#### 4.13.1.1.4 仅限 B4J

##### **File.DirApp**

返回应用程序文件夹。

##### **File.DirData**

返回适合写入文件的文件夹路径。

在 Windows 上，Program Files 下的文件夹是只读的。因此 File.DirApp 也将是只读的。  
此方法返回与非 Windows 计算机上的 File.DirApp 相同的路径。

在 Windows 上，它返回用户数据文件夹的路径。例如：

C:\Users\[用户名]\AppData\Roaming\[AppName]

##### **File.DirTemp**

返回临时文件夹。

#### 4.13.1.2 文件存在? B4A, B4i, B4J

要检查文件是否已存在, 请使用:

**File.Exists** (Dir As String, FileName As String)

如果文件存在则返回 True, 如果不存在则返回 False。

**注意:** File.Exists 不适用于 File.DirAssets !!!

#### 4.13.1.3 常用方法 B4A, B4i, B4J

File 对象包含多种写入文件和读取文件的方法。

要写入文件或读取文件, 必须打开文件。

**File.OpenOutput** (Dir As String, FileName As String, Append As Boolean)

– 打开给定文件进行输出, Append 参数指示是否将文本添加到现有文件的末尾。如果文件不存在, 则会创建该文件。

**File.OpenInput** (Dir As String, FileName As String)

– 打开文件进行读取。

**File.WriteString** (Dir As String, FileName As String, Text As String)

– 将给定文本写入新文件。

**File.ReadString** (Dir As String, FileName As String) As String

– 读取文件并将其内容作为字符串返回。

**File.WriteList** (Dir As String, FileName As String, List As List)

– 将列表中存储的所有值写入文件。如果需要, 所有值都会转换为字符串类型。每个值都将存储在单独的行中。

请注意, 如果值包含换行符, 它将保存在多行中, 当您读取它时, 它将被读取为多个项目。

**File.ReadList** (Dir As String, FileName As String) As List

– 读取文件并将每行存储为列表中的项目。

**File.WriteMap** (Dir As String, FileName As String, Map As Map)

– 获取包含键和值元素对的映射对象并将其存储在文本文件中。文件格式称为 Java 属性文件: [.properties](#) – [维基百科, 自由百科全书](#)

除非文件需要手动编辑, 否则文件格式不太重要。这种格式使手动编辑变得容易。

File.WriteMap 的一个常见用途是将“设置”映射保存到文件中。

**File.ReadMap** (Dir As String, FileName As String) As Map

– 读取属性文件并将其键/值对作为 Map 对象返回。请注意, 返回的条目顺序可能与原始顺序不同。

**File.WriteBytes** (Dir As String, FileName As String, Data As Byte())

– 将给定的文本写入新文件。

**File.ReadBytes** (Dir As String, FileName As String)



- 从给定的文件中读取数据。

返回: Byte()

**File.Copy** (DirSource As String, FileSource As String, DirTarget As String, FileTarget As String)

- 将源文件从源目录复制到目标目录中的目标文件。

请注意, 无法将文件复制到 Assets 文件夹。

**File.Copy2** (In As InputStream, Out As OutputStream)

- 将输入流中的所有可用数据复制到输出流。

输入流在结束时自动关闭。

**File.Delete** (Dir As String, FileName As String)

- 从给定目录中删除给定文件。

**File.ListFiles** (Dir As String) As List

- 列出指定目录中的文件和子目录。

示例:

```
Private List1 As List
```

```
List1 = File.ListFiles(File.DirInternal)
```

List1 可以在 Sub Globals 中声明

**File.Size** (Dir As String, FileName As String)

- 返回指定文件的大小 (以字节为单位)。

此方法不支持资源文件夹中的文件。

**File.MakeDir** (Parent As String, Dir)

- 创建给定文件夹 (根据需要创建所有文件夹)。

示例:

```
File.MakeDir(File.DirInternal, "music/90")
```

### 4.13.2 文件名

B4X 文件名允许使用以下字符：

**a** 到 **z**, **A** 到 **Z**, **0** 到 **9**, 点 **.** 下划线 **\_** 以及后面的字符 **+ - % &**

不允许使用空格和后面的字符 **\* ?**。

示例：MyFile.txt

请注意，B4X 文件名区分大小写！

**MyFile.txt** 与 **myfile.txt** 不同

### 4.13.3 子文件夹

您可以在 B4X 中定义子文件夹。

```
File.MakeDir(File.DirInternal, "Pictures")
```

要访问子文件夹，您应该将子文件夹名称添加到文件夹名称中，中间用 “/” 隔开。

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal & "/Pictures", "test1.png")
```

或者在文件名前添加子文件夹名称，中间用 “/” 隔开。

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal, "Pictures/test1.png")
```

两种方法都可行。

#### 4.13.4 B4A, B4J TextWriter

对于文本文件，还有另外两个有用的函数：**TextWriter** 和 **TextReader**：

**TextWriter.Initialize** (OutputStream As OutputStream)

- 将 TextWriter 对象初始化为输出流。

示例：

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirInternal, "Test.txt", False))
```

Writer 可以在 Sub Globals 中声明。

**TextWriter.Initialize2** (OutputStream As OutputStream , Encoding As String)

- 将 TextWriter 对象初始化为输出流。
- Encoding 表示文本编码的 CodePage（也称为 CharSet）（请参阅下一章）。

示例：

```
Private Writer As TextWriter
Writer.Initialize2(File.OpenOutput(File.DirInternal, "Test.txt", False), "ISO-8859-1")
```

Writer 可以在 Sub Globals 中声明。

参见：[文本编码](#)

**TextWriter.Write** (Text As String)

- 将给定的文本写入流。

**TextWriter.WriteLine** (Text As String)

- 将给定的文本写入流，后跟换行符 LF Chr(10)。

**TextWriter.WriteLineList** (List As List)

- 将列表中的每个项目作为一行写入。

请注意，包含 CRLF 的值将保存为两行（使用 ReadList 读取时将返回两个项目）。  
所有值都将转换为字符串。

**TextWriter.Close**

- 关闭流。

示例：

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirInternal, "Text.txt", False))
Writer.WriteLine("This is the first line")
Writer.WriteLine("This is the second line")
Writer.Close
```

### 4.13.5 B4A, B4J TextReader

对于文本文件，还有另外两个有用的函数：TextWriter 和 TextReader：

**TextReader.Initialize** (InputStream As InputStream)

– 将 TextReader 初始化为输入流。

示例：

```
Private Reader As TextReader
Reader.Initialize(File.OpenInput(File.DirInternal, "Test.txt"))
```

Reader 可以在 Sub Globals 中声明。

**TextReader.Initialize2** (InputStream As InputStream, Encoding As String)

– 将 TextReader 初始化为输入流。

– Encoding 表示 CodePage（也称为 CharSet），即文本编码。

示例：

```
Private Reader As TextReader
Reader.Initialize2(File.OpenInput(File.DirInternal, "Test.txt", "ISO-8859-1"))
```

Reader 可以在 Sub Globals 中声明。

参见：[文本编码](#)

**TextReader.ReadAll** As String

– 读取所有剩余文本并关闭流。

示例：

```
txt = Reader.ReadAll
```

**TextReader.ReadLine** As String

– 从流中读取下一行。

不返回换行符。

如果没有更多字符可读取，则返回 Null。

示例：

```
Private Reader As TextReader
Reader.Initialize(File.OpenInput(File.DirInternal, "Test.txt"))
Private line As String
line = Reader.ReadLine
Do While line <> Null
    Log(line)
    line = Reader.ReadLine
Loop
Reader.Close
```

**TextReader.ReadList** As List

– 读取剩余文本并返回一个由行填充的 List 对象。

完成后关闭流。

示例：

```
List1 = Reader.ReadList
```

### 4.13.6 文本编码

文本编码或字符编码由将给定库中的每个字符与其他字符配对的代码组成。其他术语，如字符集 (charset)，有时还有字符映射或代码页，几乎可以互换使用（来源 Wikipedia）。

Android 中的默认字符集是 Unicode UTF-8。

在 Windows 中，最常见的字符集是 ASCII 和 ANSI。

- ASCII 包含 128 个字符的定义，其中 33 个是非打印控制字符（现在大部分已过时），会影响文本和空格的处理方式。
- ANSI, Windows-1252 或 CP-1252 是拉丁字母的字符编码，在 Microsoft Windows 的旧版组件中默认使用英语和其他一些西方语言，具有 256 个定义（一个字节）。前 128 个字符与 ASCII 编码中的相同。

在西方国家，Windows 程序生成的许多文件都使用 ANSI 字符集进行编码。例如：默认情况下，Excel csv 文件，记事本文件。

但是使用记事本，文件可以使用 *UTF-8* 编码保存。

B4X 可以使用以下字符集：

- UTF-8                默认字符集
- UTF -16
- UTF - 16 BE
- UTF - LE
- US-ASCII ASCII 字符集
- ISO-8859-1 几乎相当于 ANSI 字符集
- Windows-1251 西里尔字母
- Windows-1252 拉丁字母

要读取使用 ANSI 编码的 Windows 文件，您应该使用 *Windows-1252* 字符集。

如果您需要编写用于 Windows 的文件，您也应该使用 *Windows-1252* 字符集。

Windows 和 B4X 之间的另一个区别是行尾字符：

- B4X，仅在行尾添加 LF（换行符）字符 Chr(10)。
- Windows，在行尾添加两个字符 CR（Carriage Return Chr(13)）和 LF Chr(10)。如果您需要为 Windows 编写文件，则必须自行添加 CR。

行尾符号为：

- B4X                                CRLF                Chr(10)
- Basic4PPC                        CRLF                Chr(13) & Chr(10)

要读取或写入具有不同编码的文件，您必须使用带有 Initialize2 方法的 TextReader 或 TextWriter 对象。即使读取 csv 文件也是如此。

读取 Excel csv 文件的提示：

您可以：

- 在桌面上，在 *NotePad* 或 *Notepad++* 等文本编辑器中加载 csv 文件
- 使用 *UTF-8* 编码保存文件
- 使用 *Notepad++* 使用 *UTF-8* 无 BOM 编码，看下文。

或者

- 使用 `TextReader.Initialize2` 和 “*Windows-1252*” 编码读取整个文件。
- 使用 `TextWriter.Initialize` 以标准 Android 编码将其保存回来。
- 使用 `StringUtils` 库中的 `LoadCSV` 或 `LoadCSV2` 读取文件。

```
Private txt As String
Private tr As TextReader
tr.Initialize2(File.OpenInput(File.DirAssets, "TestCSV1_W.csv"), "Windows-1252")
txt = tr.ReadAll
tr.Close
```

```
Private tw As TextWriter
tw.Initialize(File.OpenOutput(File.DirInternal, "TestCSV1_W.csv", False))
tw.Write(txt)
tw.Close
```

```
lstTest = StrUtil.LoadCSV2(File.DirInternal, "TestCSV1_W.csv", ";", lstHead)
```

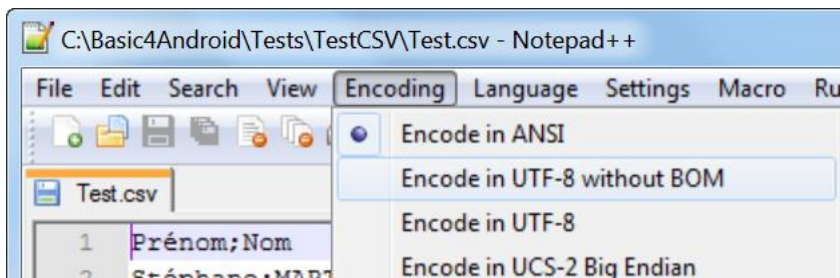
使用 *NotePad* 保存文件时，会添加三个额外字节。

这些字节称为 BOM 字符 (Byte Order Mark) (字节顺序标记)。

在 *UTF-8* 中，它们由以下字节序列表示：0xEF, 0xBB, 0xBF。

将文本解释为 *Windows-1252* 的文本编辑器或 Web 浏览器将显示字符 *ï»¿*。

为避免这种情况，您可以使用 *Notepad++* 代替 *NotePad*，并使用不带 BOM 的 *UTF-8* 编码。



将文本从 *Windows-1252* 更改为 *UTF-8* 的另一种可能性是使用以下代码。

```
Private var, result As String
var = "Gestió"
Private arrByte() As Byte
arrByte = var.GetBytes("Windows-1252")
result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")
```

## 4.14 列表 仅限 B4A, B4i 和 B4J

列表类似于动态数组。

列表 List 必须先初始化, 然后才能使用。

- **Initialize** 初始化一个空列表。

```
Private List1 As List
List1.Initialize
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

- **Initialize2** (一些数组)

用给定的值初始化一个列表。此方法应用于将数组转换为列表。请注意, 如果您将列表传递给此方法, 则两个对象将共享同一个列表, 如果您传递数组, 则列表将具有固定大小。这意味着您以后不能添加或删除元素。

示例 1:

```
Private List1 As List
List1.Initialize2(Array As Int(1, 2, 3, 4, 5))
```

示例 2:

```
Private List1 As List
Private SomeArray(10) As String
' Fill the array
List1.Initialize2(SomeArray)
```

您可以从列表中添加和删除元素, 它会相应地更改其大小。  
无论是:

- **Add (item As Object)**  
在列表末尾添加一个值。  
`List1.Add(Value)`
- **AddAll (Array As String("value1", "value2"))**  
将数组的所有元素添加到列表的末尾。  
`List1.AddAll(List2)`  
`List1.AddAll(Array As Int(1, 2, 3, 4, 5))`
- **AddAllAt (Index As Int, List As List)**  
从给定位置开始将数组的所有元素插入列表中。  
`List1.AddAll(12, List2)`  
`List1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))`
- **InsertAt (Index As Int, Item As Object)**  
在指定索引中插入指定元素。  
结果, 所有索引大于或等于指定索引的项目都会被移动。  
`List1.InsertAt(12, Value)`
- **RemoveAt (Index As Int)**  
从列表中删除给定位置的指定元素。  
`List1.RemoveAt(12)`

列表可以包含任何类型的对象。然而，如果一个列表被声明为一个进程全局对象，它就不能保存活动对象（如视图）。

B4X 自动将常规数组转换为列表。因此，当需要一个 List 参数时，您可以改为传递一个数组。

获取列表的大小：

- `List1.Size`

使用 `Get` 方法从列表中获取元素（列表索引基于 0）：

要获取第一项，请使用 `Get(0)`。

要获取最后一项，请使用 `Get(List1.Size - 1)`。

- `Get(Index As Int)`  
`number = List1.Get(i)`

您可以使用 `For` 循环遍历所有值：

```
For i = 0 To List1.Size - 1
    Private number As Int
    number = List1.Get(i)
    ...
Next
```

列表可以通过以下文件保存和加载：

- `File.WriteList(Dir As String, FileName As String, List As List)`  
`File.WriteList(File.DirRootExternal, "Test.txt", List1)`
- `File.ReadList (Dir As String, FileName As String)`  
`List1 = File.ReadList(File.DirRootExternal, "Test.txt")`

可以通过以下方式更改单个项目：

- `List1.Set(Index As Int, Item As Object)`  
`List1.Set(12, Value)`

可以使用以下方式对列表进行排序（元素必须全部为数字或字符串）：

- `Sort(Ascending As Boolean)`  
`List1.Sort(True)`                    sort ascending  
`List1.Sort(False)`                sort descending
- `SortCaseInsensitive(Ascending As Boolean)`

清除列表：

- `List1.Clear`



### 4.14.1 非动态列表

下面的代码将无法运行，会出现一个错误：

```
List1 = Array As String("Val1", "Val2", "Val3")
List1.Add("Val4")
```

此代码也不会起作用：

```
List1.Initialize2(Array As String("Val1", "Val2", "Val3"))
List1.Add("Val4")
```

因为上面的初始化会生成非动态列表，这些列表无法更改。

请注意，如果您想复制列表，下面的代码也将不起作用：

```
Private List1 As List
List1.Initialize
List1.AddAll(Array As String("Val1", "Val2", "Val3"))
Private List2 As List
List2 = List1
Log(List1.Size)
Log(List2.Size)
List1.Add("Val4")
Log(List1.Size)
Log(List2.Size)
```

日志显示：

```
3
3
4
4
```

您会发现，当您修改 List1 中的某些内容时，List2 中的内容也会被修改。这是设计使然，列表是通过引用传递的。

要获得 List 的独立副本，您需要将：

```
List2 = List1
```

替换为

```
List2.Initialize
List2.AddAll(List1)
```

就像下面的代码：

```
Private List1 As List
List1.Initialize
List1.AddAll(Array As String("Val1", "Val2", "Val3"))
Private List2 As List
List2.Initialize
List2.AddAll(List1)
Log(List1.Size)
Log(List2.Size)
List1.Add("Val4")
Log(List1.Size)
Log(List2.Size)
```

日志显示：

```
3
3
4
3
```

您会看到 List2 的大小没有改变。

## 4.15 地图 仅限 B4A, B4i 和 B4J

地图 Map 是一个包含键和值对的集合。

钥匙是独一无二的。这意味着如果您添加一个键/值对（条目）并且该集合已经包含具有相同键的条目，则先前的条目将从映射中删除。

键应该是字符串或数字。该值可以是任何类型的对象。

与列表类似，地图可以保存任何对象，但是如果它是流程全局变量，则它不能保存活动对象（如视图）。

地图对于存储应用程序设置非常有用。

本例中使用了地图：

- DBUtils 模块  
用于数据库条目，键是列名，值是列值。

Map 必须先初始化，然后才能使用。

- Initialize 初始化一个空地图。  
`Private Map1 As Map`  
`Map1.Initialize`

添加一个新条目：

- Put(Key As Object, Value As Object)  
`Map1.Put("Language", "English")`

获取条目：

- Get(Key As Object)  
`Language = Map1.Get("Language")`

获取给定索引处的键或值（仅 B4A 和 B4J）：

返回给定索引处元素的值。

GetKeyAt 和 GetValueAt 应该用于遍历所有元素。

这些方法针对按升序迭代项目进行了优化。

- GetKeyAt(Index As Int)  
`Key = Map1.GetKeyAt(12)`

获取给定索引处的值（仅 B4A 和 B4J）：

- GetValueAt(Index As Int)  
`Value = Map1.GetValueAt(12)`

检查 Map 是否包含条目，测试是否存在具有给定键的条目：

- ContainsKey(Key As Object):  
`If Map1.ContainsKey("Language") Then`  
`Msgbox("There is already an entry with this key !", "ATTENTION")`  
`Return`  
`End If`

删除一个条目:

- `Remove(Key As Object)`  
`Map1.Remove("Language")`

清除, 清除地图中的所有元素:

- `Clear`  
`Map1.Clear`

地图可以保存和加载:

- `File.WriteMap(Dir As String, FileName As String, Map As Map)`  
`File.WriteMap(File.DirInternal, "settings.txt", mapSettings)`
- `ReadMap(Dir As String, FileName As String)`  
读取文件并将每一行解析为键值对 (字符串)。  
请注意, 地图中项目的顺序可能与文件中的顺序不同。  
`mapSettings = File.ReadMap(File.DirInternal, "settings.txt")`
- `File.ReadMap2(Dir As String, FileName As String, Map As Map)`  
类似于 `ReadMap`。 `ReadMap2` 将项目添加到给定的 `Map`。  
通过使用带有填充地图的 `ReadMap2`, 您可以根据需要强制项目排序。  
`mapSettings = File.ReadMap2(File.DirInternal, "settings1.txt", mapSettings)`

## 4.16 类模块

在 B4X 中，您可以使用三种类型的类模块：

- Standard Class modules                      标准类
- B4XPages    B4XPages
- CustomView Class Modules                  专用于自定义视图的

在本章中，我们将仅看到标准类模块。

B4XPages 在 [B4XPages 跨平台项目](#) 手册中进行了说明。

CustomView 类模块在 [B4X CustomViews](#) 手册中进行了说明。

### 4.16.1 入门

来自 [Wikipedia](#) 的类别定义：

在面向对象编程中，类是一种用于创建自身实例的构造 - 称为类实例，类对象，实例对象或简称为对象。类定义组成成员，使其实例具有状态和行为。数据字段成员（成员变量或实例变量）使类实例能够保持状态。其他类型的成员，尤其是方法，使类实例能够具有行为。类定义其实例的类型。

类通常表示名词，例如人，地点或事物，或名词化的东西。例如，“香蕉”类将表示香蕉的一般属性和功能。单个特定的香蕉将是“香蕉”类的一个实例，即“香蕉”类型的对象。

我们先从一个例子开始，源代码：/Person 文件夹中的 *SourceCode\Person*。

在 Person 模块中

```
'人员模块
Sub Class_Globals
    Private FirstName, LastName As String
    Private BirthDate As Long
End Sub

Sub Initialize (aFirstName As String, aLastName As String, aBirthDate As Long)
    FirstName = aFirstName
    LastName = aLastName
    BirthDate = aBirthDate
End Sub

Public Sub GetName As String
    Return FirstName & " " & LastName
End Sub

Public Sub GetCurrentAge As Int
    Return GetAgeAt(DateTime.Now)
End Sub

Public Sub GetAgeAt(Date As Long) As Int
    Private diff As Long
    diff = Date - BirthDate
    Return Floor(diff / DateTime.TicksPerDay / 365)
End Sub
```

主模块。

```
Sub Activity_Create(FirstTime As Boolean)
    Private p As Person
    p.Initialize("John", "Doe", DateTime.DateParse("05/12/1970"))
    Log(p.GetCurrentAge)
End Sub
```

我将首先解释类，代码模块和类型之间的区别。

与类型类似，类是模板。从此模板，您可以实例化任意数量的对象。

类型字段类似于类全局变量。但是，与仅定义数据结构的类型不同，类还定义行为。行为在类的子类中定义。

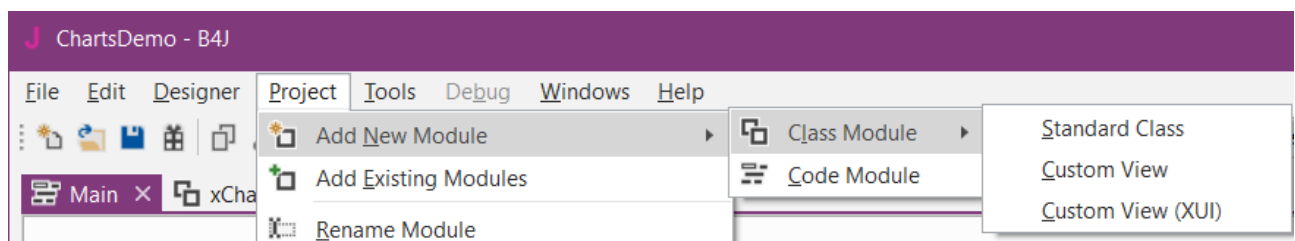
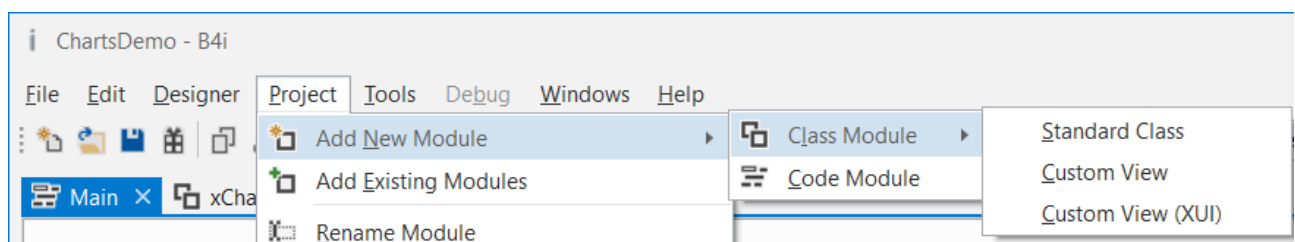
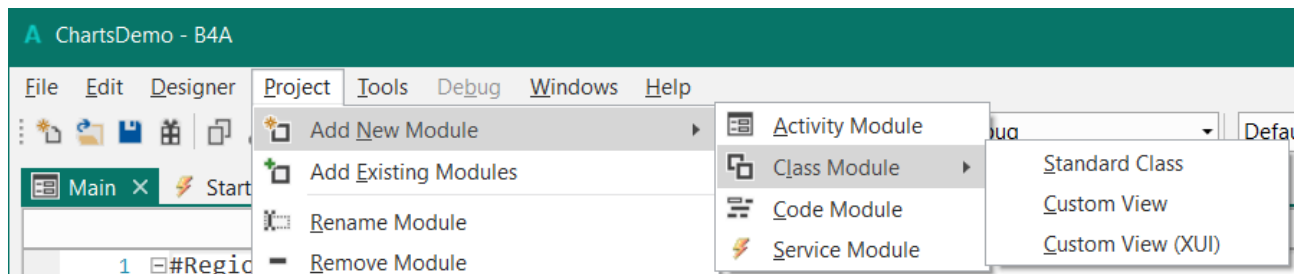
与作为对象模板的类不同，代码模块是子类的集合。代码模块和类之间的另一个重要区别是，代码模块始终在调用子类的上下文中运行。代码模块不保存对任何上下文的引用。因此，无法处理事件或将 CallSub 与代码模块一起使用。

类存储对调用 Initialize 子类的模块上下文的引用。这意味着类对象与初始化它们的模块共享相同的生命周期。

### 4.16.1.1 添加类模块

通过选择“项目”>“添加新模块”>“类模块”或“添加现有模块”，可以添加新的或现有的类模块。

与其他模块一样，类保存为带有 *bas* 扩展名的文件。

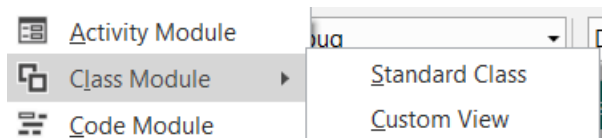


有两种类模块类型：

[Standard Class](#)

CustomView

CustomView (XUI)

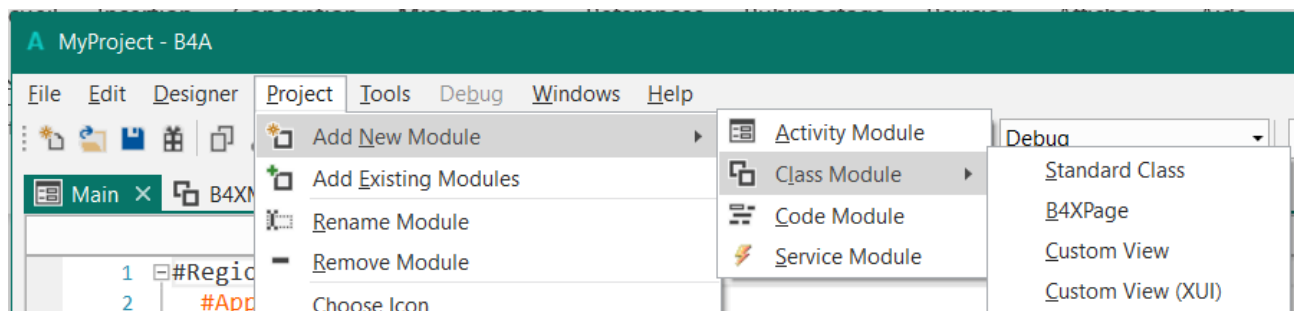


☒ Core (Version: 8.30)

☒ XUI (Version: 1.72)

仅当选择 XUI 库时才会显示 CustomView (XUI)！

如果您使用 B4XPages 模板，您可以选择 B4XPage 来创建 B4XPage 类。



### 4.16.1.2 多态性

多态性(Polymorphism)允许您以相同的方式处理遵循相同接口的不同类型的对象。

B4X 多态性类似于 Duck 类型 [Duck typing](#) 概念。

作为示例，我们将创建两个名为 Square 和 Circle 的类。

每个类都有一个名为 Draw 的子类，用于将对象绘制到画布上：

Draw 文件夹中的源代码 *Draw*。

以下代码是 B4A 代码。

'类 Square 模块

Sub Class\_Globals

Private mx, my, mWidth As Int

End Sub

'初始化对象。如果需要，您可以向此方法添加参数。

Sub Initialize (Shapes As List, x As Int, y As Int, length As Int)

mx = x

my = y

mLength = length

Shapes.Add(Me)

End Sub

Sub Draw(c As Canvas)

Private r As Rect

r.Initialize(mx, my, mx + mLength, my + mLength)

c.DrawRect(r, Colors.Red, False, 1dip)

End Sub

'类 Circle 模块

Sub Class\_Globals

Private mx, my, mRadius As Int

End Sub

'初始化对象。如果需要，您可以向此方法添加参数。

Sub Initialize (Shapes As List, x As Int, y As Int, radius As Int)

mx = x

my = y

mRadius = radius

Shapes.Add(Me)

End Sub

Sub Draw(cvs As Canvas)

cvs.DrawCircle(mx, my, mRadius, Colors.Blue, False, 1dip)

End Sub

在主模块中，我们创建一个包含正方形和圆形的 **Shapes** 列表。然后我们遍历列表并绘制所有对象：

```
Sub Process_Globals
    Public Shapes As List
End Sub

Sub Globals
    Private cvs As Canvas
End Sub

Sub Activity_Create(FirstTime As Boolean)
    cvs.Initialize(Activity)
    Private Square1, Square 2 As Square
    Private Circle1 As Circle
    Shapes.Initialize
    Square1.Initialize(Shapes, 110dip, 110dip, 50dip)
    Square2.Initialize(Shapes, 10dip, 10dip, 100dip)
    Circle1.Initialize(Shapes, 50%x, 50%y, 100dip)

    DrawAllShapes
End Sub

Sub DrawAllShapes
    For i = 0 To Shapes.Size - 1
        CallSub2(Shapes.Get(i), "Draw", cvs)
    Next
    Activity.Invalidate
End Sub
```

如您所见，我们不知道列表中每个对象的具体类型。我们只是假设它有一个 **Draw** 方法，该方法需要一个 **Canvas** 参数。稍后我们可以轻松添加更多类型的形状。您可以使用 **SubExists** 关键字来检查对象是否包含特定子对象。您还可以使用 **Is** 关键字来检查对象是否属于特定类型。



### 4.16.1.3 自引用

Me 关键字返回对当前对象的引用。Me 关键字只能在类模块内部使用。

考虑上面的例子。我们将 Shapes 列表传递给 Initialize 子程序，然后将每个对象从 Initialize 子程序添加到列表中：

```
Sub Initialize (Shapes As List, x As Int, y As Int, radius As Int)
    mx = x
    my = y
    mRadius = radius
    Shapes.Add(Me)
End Sub
```

### 4.16.1.4 活动对象 仅限 B4A

这一点与 Android 活动特殊生命周期有关。

请务必先阅读[活动和进程生命周期教程](#)。

Android UI 元素保存对父活动的引用。由于操作系统被允许终止后台活动以释放内存，因此不能将 UI 元素声明为进程全局变量（这些变量与进程一样长）。此类元素称为 Activity 对象。自定义类也是如此。如果一个或多个类全局变量属于 UI 类型（或任何活动对象类型），则该类将被视为“活动对象”。这意味着不能将此类的实例声明为进程全局变量。

## 4.16.2 标准类模块

### 4.16.2.1 结构

标准类的默认模板：

B4A 和 B4i

```
Sub Class_Globals
```

```
End Sub
```

```
'Initializes the object. You can add parameters to this method if needed.
```

```
Public Sub Initialize
```

```
End Sub
```

B4J

```
Sub Class_Globals
```

```
    Private fx As JFX
```

```
End Sub
```

```
'Initializes the object. You can add parameters to this method if needed.
```

```
Public Sub Initialize
```

```
End Sub
```

仅预定义了两个例程：

**Sub Class\_Globals** - 此子类与 Main Globals 子类类似。这些变量将是类全局变量（有时称为实例变量或实例成员）。

在 B4J 中，默认声明 fx 库。如果不需要，您可以将其删除。

**Sub Initialize** - 必须先初始化类对象，然后才能调用任何其他子程序。通过调用 Initialize 子程序来初始化对象。调用 Initialize 时，可以设置对象的上下文（父对象或服务）。

请注意，您可以修改此子签名并根据需要添加参数。

示例：Person 类模块

源代码位于 Person 文件夹中。

所有三个 B4X 平台（B4A, B4i, B4J）的代码均相同。

```
'Class Person module
Sub Class_Globals
    Private mFirstName, mLastName As String
    Private mBirthDate As Long
End Sub

Sub Initialize (FirstName As String, LastName As String, BirthDate As Long)
    mFirstName = FirstName
    mLastName = LastName
    mBirthDate = BirthDate
End Sub

Public Sub GetName As String
    Return mFirstName & " " & mLastName
End Sub

Public Sub GetCurrentAge As Int
    Return GetAgeAt(DateTime.Now)
End Sub

Public Sub GetAgeAt(Date As Long) As Int
    Dim diff As Long
    diff = Date - mBirthDate
    Return Floor(diff / DateTime.TicksPerDay / 365)
End Sub
```

在上面的代码中，我们创建了一个名为 Person 的类，然后在主模块中实例化该类型的对象：

```
Private p As Person
p.Initialize("John", "Doe", DateTime.Parse("05/12/1970"))
Log(p.GetCurrentAge)
```

如果对象本身已经初始化，则不需要调用初始化：

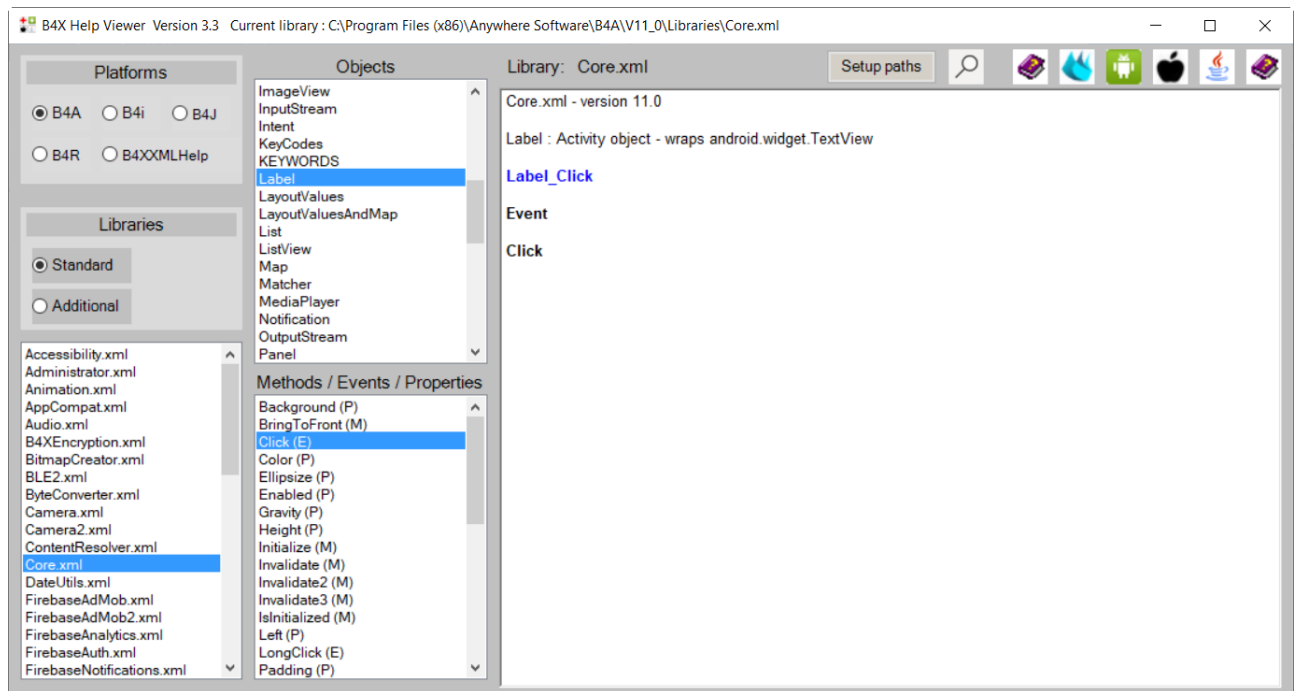
```
Private p2 As Person
p2 = p '两个变量现在指向同一个 Person 对象。
Log(p2.GetCurrentAge)
```

## 5 查找对象方法, 属性, 事件

### 5.1 B4X 帮助查看器

B4X 帮助工具手册中详细介绍了 B4X 帮助查看器。

您可以选择一个平台、一个库、一个对象并显示主题。

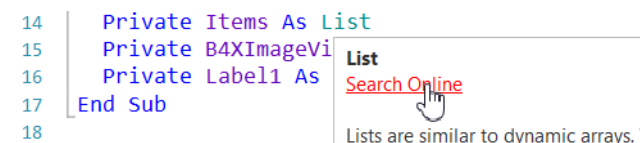
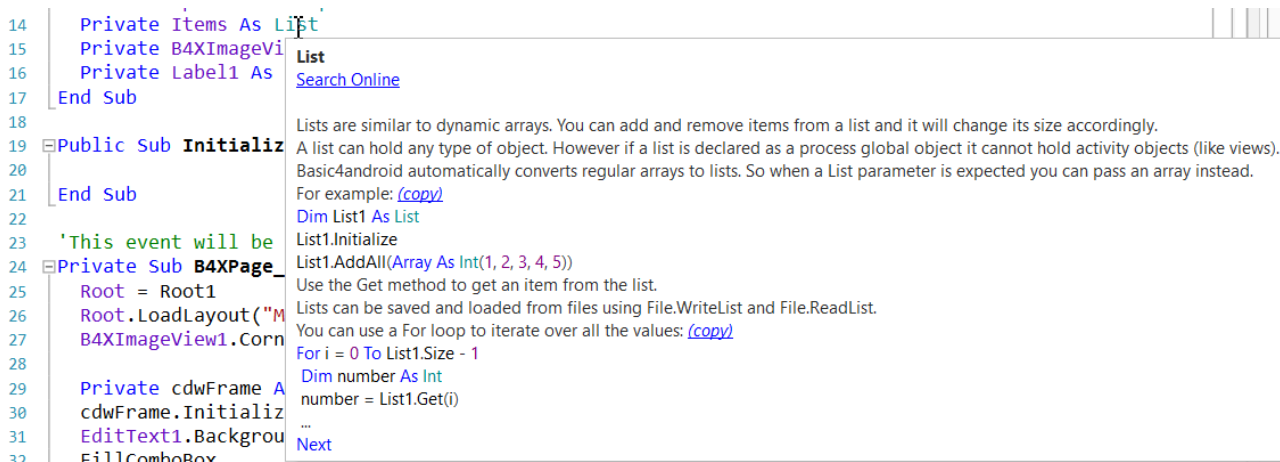


可以通过以下链接从论坛下载:

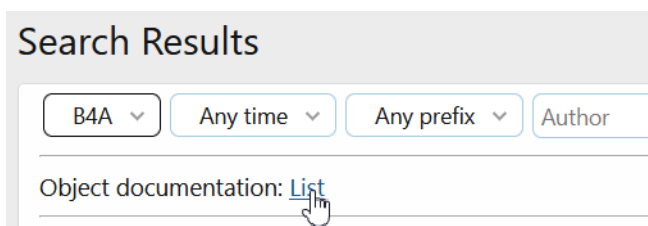
<https://www.b4x.com/android/forum/threads/b4x-help-viewer.46969/>

## 5.2 悬停在对象上

在代码中, 将鼠标悬停在某个对象上就会显示内联帮助, 示例中为一个列表。



当您将鼠标悬停在“在线搜索”上并点击:



您可以在论坛中看到此页面, 将鼠标悬停在 List 上.

而结果.

### List

Lists are similar to dynamic arrays. You can add and remove items from a list and it will change its size accordingly. A list can hold any type of object. However if a list is declared as a process global object it cannot hold activity objects (like views). Basic4android automatically converts regular arrays to lists. So when a List parameter is expected you can pass an array instead. For example:

```
Dim List1 As List
List1.Initialize
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

Use the Get method to get an item from the list.

Lists can be saved and loaded from files using **File.WriteList** and **File.ReadList**.

You can use a For loop to iterate over all the values:

```
For i = 0 To List1.Size - 1
    Dim number As Int
    number = List1.Get(i)
    ...
Next
```

### Events:

None

### Members:

- [Add](#) (item As Object)
- [AddAll](#) (List As List)
- [AddAllAt](#) (Index As Int, List As List)
- [Clear](#)

## 5.3 定义事件例程

在代码中输入 Private Sub 或 Sub 和一个空格:

```
86 |  
87 | Private Sub |  
    | Press Tab to insert event declaration.
```

然后按 Tab, 您将获得项目中可能的所有对象的列表, 包括所选库的对象。

```
86 |  
87 | Private Sub | Select type and press enter  
    |  
    | Activity  
    | AutoCompleteEditText  
    | B4XBreadCrumb  
    | B4XComboBox  
    | B4XFloatTextField
```

选择一个对象, 示例中的 Activity:

```
86 |  
87 | Private Sub | Select type and press enter  
    |  
    | Activity  
    | AutoCompleteEditText  
    | B4XBreadCrumb  
    | B4XComboBox  
    | B4XFloatTextField
```

选择事件:

```
86 |  
87 | Private Sub | Select type and press enter Activity >  
    |  
    | ActionBarHomeClick  
    | Click  
    | KeyPress (KeyCode As Int) As Boolean 'Return True to consume the event  
    | KeyUp (KeyCode As Int) As Boolean  
    | LongClick  
    | PermissionResult (Permission As String, Result As Boolean)  
    | Touch (Action As Int, X As Float, Y As Float)  
    | WindowFocusChanged (Focused As Boolean)
```

输入对象名称并按回车键。

```
87 | Private Sub EventName_Touch (Action As Int, X As Float, Y As Float)  
88 |  
89 | End Sub
```

而结果:

```
87 | Private Sub Activity_Touch (Action As Int, X As Float, Y As Float)  
88 |  
89 | End Sub
```

## 6 代码片段

您可以将代码片段添加到 IDE。

代码片段是一段代码，只需单击几下即可将其添加到您的代码中。

代码片段可以包含任意数量的变量或占位符，这些变量或占位符将在插入代码片段后立即突出显示和编辑。

这些是带有代码的简单文本文件。

您可以在代码片段中使用任意数量的变量。这些变量位于两个 \$ 字符之间，如 \$我的变量\$。

复制代码片段时，这些变量将被复制而不带 \$ 字符并突出显示。

在 *Code snippet in a b4xlib library* 中显示了一个示例。

有一个保留变量名 \$end\$，当复制代码片段时，它会将光标移动到此位置。

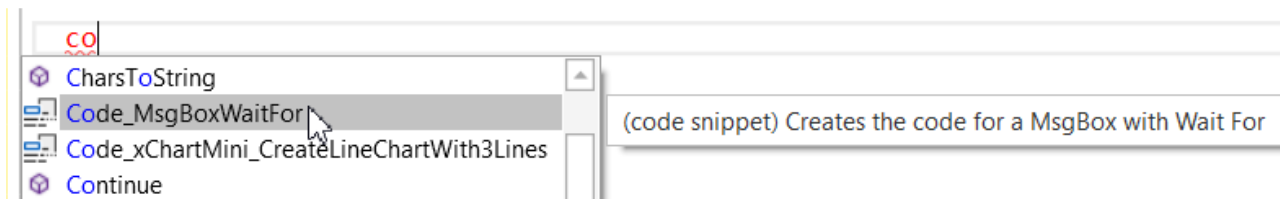
在 *AdditionalLibraries\B4X\Snippets* 文件夹中的简单代码片段一章中显示了一个示例。

### 6.1 在 IDE 中复制代码片段

在 IDE 中输入“co”作为代码。

所有代码片段都以 Code\_ 开头。

单击它们时，您将看到注释（如果有）。



在上面的例子中，您会看到两个代码片段。

- **Code\_MsgBoxWaitFor**  
位于 *AdditionalLibraries\B4X\Snippets* 文件夹中的代码。
- **Code\_xChartMini\_CreateLineChartWith3Lines**  
位于 *xChartMini b4xlib* 库中的代码片段。

当您单击 **Code\_MsgBoxWaitFor** 时，代码片段将被复制到编辑器中。

```
Private sf As Object = xui.Msgbox2Async("Message", "Title", "Positive", "Cancel", "Negative", Null)
Wait For (sf) Msgbox_Result (Result As Int)
If Result = xui.DialogResponse_Positive Then
    |
End If
```

## 6.2 代码片段示例

### 6.2.1 AdditionalLibraries\B4X\Snippets 文件夹中的简单代码片段

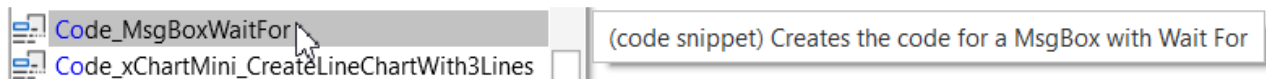
让我们为具有 Wait / For 结构的 MessageBox 制作一个代码片段。

代码：

```
'Creates the code for a MsgBox with Wait For
Private sf As Object = xui.Msgbox2Async("Message", "Title", "Positive", "Cancel",
"Negative", Null)
Wait For (sf) Msgbox_Result (Result As Int)
If Result = xui.DialogResponse_Positive Then
$end$
End If
```

我们将代码保存在 AdditionalLibraries\B4X\Snippets 文件夹中的 txt 文件中，文件名为 MsgBoxWaitFor.txt。

该文件的名称就是您在 IDE 中看到的 *Code\_*前缀后的内容。



IDE 中（代码片段）后面的注释是代码中第一个注释行，此行不会被复制。

在代码的倒数第二行中，您会看到 \$end\$。

这是一个特殊的词，当复制代码片段时，它会将光标直接设置在其位置，看下文。

```
Private sf As Object = xui.Msgbox2Async("Message", "Title", "Positive", "Cancel", "Negative", Null)
Wait For (sf) Msgbox_Result (Result As Int)
If Result = xui.DialogResponse_Positive Then
|
End If
```



## 6.2.2 b4xlib 库中的代码片段

这是 xChartMini b4xlib 库创建折线图的代码片段。

代码：

```
'Create a line chart with 3 lines
Private Sub CreateLineChart
    ' clear previous data
    $xChart1$.ClearData
    ' initialize the line data
    $xChart1$.AddLine("Random", xui.Color_Blue)
    $xChart1$.AddLine("Cos", xui.Color_Red)
    $xChart1$.AddLine("Sin", xui.Color_Magenta)
    ' set the max and min scale values
    $xChart1$.YScaleMaxValue = 10
    $xChart1$.YScaleMinValue = 0
    ' Add the line points.
    Dim Val1, Val2, Val3 As Double
    For i = 0 To 720 Step 15
        ' In the case of 2 lines or more we are adding an array of values.
        ' One for each line.
        Val1 = Rnd(-100, 101) / 50 + 5
        Val2 = 3 * CosD(i) + 5
        Val3 = 4 * SinD(i) + 5
        $xChart1$.AddLineMultiplePoints(i, Array As Double(Val1, Val2, Val3), i Mod 90 = 0)
    Next
    ' draw the chart
    $xChart1$.DrawChart
End Sub
```

在代码中我使用了一个变量 `$xChart1$`，它用作代码中 xChart 对象的占位符。

当用户复制代码片段时，`$xChart1$` 将突出显示，用户可以更改它。当他更改第一个出现的位置时，所有其他位置将自动更新。

代码以文本文件的形式保存在 xChartMini b4xlib 库的 Snippets 文件夹中，名称为 CreateLineChartWith3Lines。

在 IDE 中我们看到：

Code\_  
xChartMini  
CreateLineChartWith3Lines  
Create a line chart with 3 lines

这是代码片段前缀。  
这是 b4xlib 库的名称。  
这是代码片段的文件名。  
这是注释，它是上面代码片段中的第一行。



当我们点击：



代码被复制到 IDE 中，并且变量 xChart1 在此例中以红色突出显示，因为 xChart1 尚未声明。

```
Private Sub CreateLineChart
    ' clear previous data
    xChart1.ClearData

    ' initialize the line data
    xChart1.AddLine("Random", xui.Color_Blue)
    xChart1.AddLine("Cos", xui.Color_Red)
    xChart1.AddLine("Sin", xui.Color_Magenta)
```

当我们将 xChart1 更改为 LineChart1 时，所有 xChart1 变量都会更新。当然，LineChart1 已在 Class\_Globals 例程中声明。

```
Private Sub CreateLineChart
    ' clear previous data
    LineChart1.ClearData

    ' initialize the line data
    LineChart1.AddLine("Random", xui.Color_Blue)
    LineChart1.AddLine("Cos", xui.Color_Red)
    LineChart1.AddLine("Sin", xui.Color_Magenta)
```

xChart1 被 LineChart1 替换，LineChart1 的颜色正常，因为该变量已在测试项目中声明。

如果新变量尚未声明，则新变量将保持为红色。

```
Private Sub CreateLineChart
    ' clear previous data
    LineChart2.ClearData

    ' initialize the line data
    LineChart2.AddLine("Random", xui.Color_Blue)
    LineChart2.AddLine("Cos", xui.Color_Red)
    LineChart2.AddLine("Sin", xui.Color_Magenta)
```

## 6.3 代码片段的位置

### 6.3.1 在附加库文件夹下的特殊 Snippets 子文件夹中

带有 Snippets 子文件夹的 AdditionalLibraries 文件夹结构：

▼	AdditionalLibraries	
	B4A	B4A 附加库的文件夹。
	B4i	B4i 附加库的文件夹。
	B4J	B4J 附加库的文件夹。
>	B4R	B4R 附加库的文件夹。
▼	B4X	B4X 库的文件夹。
	Snippets	代码片段的文件夹。B4X 文件夹中的子文件夹。
	B4XlibXMLFiles	B4X 库 XML 文件的文件夹。

### 6.3.2 在 b4xlib 库中

您可以通过将代码片段文本文件添加到 b4xlib zip 文件中的 Snippets 文件夹中，在 b4xlib 库中添加代码片段。

b4xlib 库的结构在 [B4X 库 \\*.b4xlib](#) 章中进行了说明。

## 7 应避免的“代码异味”代码

“代码异味”是一种常见模式，可以表明代码中存在问题。问题并不意味着代码无法工作，可能是代码难以维护，或者有更优雅的方法来实现相同的功能。请记住，并非所有事情都是明确的，任何规则都有例外。

### 7.1 初始化一个对象，然后将不同的对象赋给同一个变量

```
'不好
Dim List1 As List
List1.Initialize '<-- 此处创建了一个新列表
List1 = SomeOtherList '<--- 先前的列表已被替换

'好
Dim List1 As List = SomeOtherList
```

### 7.2 已弃用的方法 - DoEvents, MsgBox

这些方法已被弃用，因此您不应再使用这些方法。

更多信息请见：

<https://www.b4x.com/android/forum/t...cated-and-async-dialogs-msgbox.79578/#content>

### 7.3 已弃用的方法 - Map.GetKeyAt / GetValueAt

已弃用的方法 - Map.GetKeyAt / GetValueAt - 这些方法是在 For Each 循环可用之前添加的。它们不是跨平台的，也不是处理地图的正确方法。

```
'不好
For i = 0 To Map1.Size - 1
    Dim key As String = Map1.GetKeyAt(i)
    Dim value As String = Map1.GetValueAt(i)
Next

'好
For Each key As String In Map1.Keys
    Dim value As String = Map1.Get(key)

Next
```

## 7.4 File.DirDefaultExternal - 这始终是一个错误

File.DirDefaultExternal - 这始终是个错误。在大多数情况下，正确的文件夹应该是 XUI.DefaultFolder (=File.DirInternal)。如果您确实需要使用外部存储，请使用 RuntimePermissions.GetSafeDirDefaultExternal。

File.DirRootExternal - 它很快就会变得无法直接访问。如果确实需要，请使用 ContentChooser 或 ExternalStorage。

## 7.5 不使用参数化查询

对于数据库查询，使用参数化查询。

'非常糟糕

```
SQL.ExecNonQuery("INSERT INTO table1 VALUES ('" & EditText1.Text & "'")
```

'丑陋，如果文本中有撇号，则会中断，并且容易受到 SQL 注入攻击。

'非常好

```
SQL.ExecNonQuery2("INSERT INTO table1 VALUES (?)", Array(EditText1.Text))
```

## 7.6 使用 Cursor 代替 ResultSet-Cursor

对于数据库查询，请使用 ResultSet 而不是 Cursor。

Cursor 是 B4A 专用对象。ResultSet 使用起来更简单一些，而且是跨平台的。

'好

```
Dim rs As ResultSet = SQL.ExecQuery2(...)
Do While rs.NextRow
    ...
Loop
rs.Close
```

## 7.7 以编程方式构建完整布局

以编程方式构建完整布局。这在 B4J 和 B4i 中尤其错误，因为存在调整大小事件，而且如果您想构建跨平台解决方案。布局可以非常轻松地移植。

## 7.8 重复代码

这个有很多种模式，但它们都很糟糕。

```
'糟糕
If b = False Then
    Button1.Text = "disabled"
    Button2.Text = "disabled"
    Button3.Text = "disabled"
    Button1.Enabled = False
    Button2.Enabled = False
    Button3.Enabled = False
Else
    Button1.Text = "enabled"
    Button2.Text = "enabled"
    Button3.Text = "enabled"
    Button1.Enabled = True
    Button2.Enabled = True
    Button3.Enabled = True
End If

'好
For Each btn As Button In Array(Button1, Button2, Button3)
    btn.Enabled = b
    If b Then btn.Text = "enabled" Else btn.Text = "disable"
Next
```

## 7.9 不使用智能字符串的长字符串

更多信息: <https://www.b4x.com/android/forum/threads/50135/#content>

```
'糟糕
Dim s As String = "This is the " & QUOTE & "first" & QUOTE & "line" & CRLF & _
    "and this is the second one. The time is " & DateTime.Time(DateTime.Now) & "."

'好
Dim s As String = $"This is the "first" line
and this is the second one. The time is $Time{DateTime.Now}.$"
```

## 7.10 在不需要时使用全局变量

```
'糟糕
Job.Initialize(Me, "") '全局变量
...

'好
Dim job As HttpJob
job.Initialize(Me, "")
```

## 7.11 当可用时不使用 Wait For

可用时却不使用 Wait For。JobDone 就是一个很好的例子: [B4X] OkHttpUtils2 with Wait For

## 7.12 使用代码模块而不是类

B4A 中的代码模块非常有限。在大多数情况下，您应该使用类而不是代码模块。代码模块是类的单个实例。

## 7.13 理解布尔值

```
'不优雅
Dim result As Boolean = DoingSomethingThatReturnTrueOrFalse
If result = True Then
    Return True
Else
    Return False
End If

' 优雅
Return DoingSomethingThatReturnTrueOrFalse
```

## 7.14 将“随机”字节转换为字符串

使用 `BytesToString` 转换为字符串的唯一有效原始字节是代表文本的字节。在所有其他情况下，转换为字符串都是错误的。即使它似乎有效，但在其他情况下也会失败。

如果您认为使用原始字节更复杂，那么您不熟悉有用的 `B4XBytesBuilder` 对象：

<https://www.b4x.com/android/forum/threads/b4x-b4xcollections-more-collections.101071/#content>

## 7.15 手动生成或解析 XML/JSON

手动生成或解析 XML / JSON。这些格式绝非易事，而且有各种没人记得的极端情况。

```
'糟糕
Dim s As String = "{"version":""," & Version &
""", "colors": ["red", "green", "blue"]}"

'好
Dim jg As JSONGenerator
jg.Initialize(CreateMap("colors": Array("red", "green", "blue"), "version": Version))
Log(jg.ToPrettyString(4))
```

## 7.16 假设地图保持秩序

与列表或数组不同，(哈希)映射通常不保留顺序。这是哈希映射工作方式所固有的，并且适用于所有编程语言。B4X 中有一些例外：- B4A 和 B4J 常规映射确实保留了顺序。- B4XCollections 中的 B4XOrderedMap 构建为列表 + 映射，它保留了顺序。它实际上允许对键进行排序，这在某些情况下很有用。常见的混淆情况是 JSON 对象 (= Map) 解析和生成。JSON 规范明确指出“不为名称/值对的排序赋予任何意义”。



## 8 Erel 建议避免的功能

B4X 和底层平台都发生了许多变化。我将尝试在此列出所有具有更好替代方案的（旧）功能。B4X 向后兼容，因此这些功能仍然有效。这些建议更适用于新项目或实施新功能时。

1. (B4A) **ListView** ➤ **xCustomListView**。  
ListView 操作起来很困难，无法自定义。它也不是跨平台的。
2. (B4i) **TableView** ➤ **xCustomListView**: 同上。
3. **CustomListView 模块** ➤ **xCustomListView 库**。  
使用该模块将会破坏其他库。
4. **Sub JobDone** ➤ **Wait For (j) JobDone**。  
[\[B4X\] OkHttpUtils2 with Wait For](#)
5. **Sub Sntp\_MessageSent (和其他)** ➤ **Wait For ...**  
<https://www.b4x.com/android/forum/threads/b4x-net-library-ftp-sntp-pop-with-wait-for.84821/#content>
6. **DoEvents / MsgBox** ➤ [DoEvents 已弃用和异步对话框 \(msgbox\)](#)
7. **各类自定义对话框** ➤ **B4XDialogs**。  
B4XDialogs 是跨平台的并且完全可定制。  
[\[B4X\] 分享您的 B4XDialog + 模板主题代码](#)
8. **File.DirDefaultExternal** ➤ **RuntimePermissions.GetSafeDirDefaultExternal**。  
<https://www.b4x.com/android/forum/threads/67689/#content>
9. **File.DirRootExternal** ➤ **ContentChooser / SaveAs**。  
<https://www.b4x.com/android/forum/threads/132731/#content>
10. **File.DirInternal / DirCache / DirLibrary / DirTemp / DirData** ➤ **XUI.DefaultFolder**
11. **Round2** ➤ **NumberFormat, B4XFormatter**  
Round2 的大部分用途是格式化数字。修改数字并不是正确的方法。
12. **带有网络流的 TextReader / TextWriter** ➤ **AsyncStreams**  
尝试在主线程上实现网络通信总是会导致糟糕的结果。
13. **TextReader / TextWriter** ➤ **File.ReadString / ReadList**  
两个例外 - 非 UTF8 文件或大文件（与 B4J 更相关）。

**14. Activities ➤ B4XPages**

这是一个很大的变化，也是最重要的变化。很难解释 B4XPages 让事情变得多么简单。项目越复杂，使用 B4XPages 就越重要。在构建非跨平台项目时也是如此。[\[B4X\] \[B4XPages\] 它到底解决了什么问题？](#)

**15. 特定于平台的 API ➤ 跨平台 API。**

当存在跨平台 API 时，这当然是相关的。一些开发人员误以为跨平台功能与平台特定的 API 相比有缺点。

- 节点 / 窗格 / 按钮 / ... ➤ B4XView
- 画布 ➤ B4XCanvas
- 各种平台特定的自定义视图 ➤ 跨平台自定义视图（例如 XUI 视图）。
- EditText / TextField / TextArea / TextView ➤ B4XFloatTextField
- fx（和其他）➤ XUI

**16. CallSubDelayed 表示可恢复子任务已完成 ➤ 作为 ResumableSub。**

[B4X\] 返回值的可恢复子任务 \(ResumableSub\)](#)

**17. CallSubDelayed / CallSubPlus 稍后再做一些事情 ➤ Sleep(x)。****18. 多种布局变体 ➤ 锚点 + 设计器脚本。**

Android 刚发布时，屏幕尺寸非常少。现在情况已经不同了。您应该构建适合任何屏幕尺寸的灵活布局。使用锚点 + 设计器脚本更容易实现。维护多个变体很困难。

**19. 以编程方式构建布局 ➤ 尽可能使用设计器。**

如果您仅使用 B4A 进行开发，那么以编程方式构建布局是一个错误，但不是一个大错误。

B4J 和 B4i 处理屏幕大小调整的方式不同，并且以编程方式处理更改要困难得多（有相关视频教程）。

大多数自定义视图只能使用设计器添加（有解决方法允许以编程方式添加它们）。

在不同平台和项目之间复制和粘贴设计器布局非常简单。

**20. 带连接的多行字符串 ➤ 智能字符串。**

[\[B4X\] Smart String Literal](#)

**21. (SQL) Cursor ➤ ResultSet。**

ResultSet 是跨平台的并且使用起来也更简单一些。

**22. ExecQuery（非参数化查询）➤ ExecQuery2。**

进行非参数化查询确实不可接受。有关更多信息，请参阅第 5 点：

<https://www.b4x.com/android/forum/t...ommon-mistakes-and-other-tips.116651/#content>

ExecNonQuery 也是如此

**23. 当可能没有结果时，使用 ExecQuerySingleResult ► ExecQuery2。**

这是一个历史性的设计错误。Null 和 String 不能混用。如果 ExecQuerySingleResult 有可能不返回任何结果 (=Null)，那么就不要再使用它，而应该使用 ExecQuery2。

**24. 使用 OkHttpUtils2 (=iHttpUtils2) 以外的任何其他库或源下载/发出 http 请求 ► OkHttpUtils2。**

OkHttpUtils2 功能非常强大，可以进行很多扩展，不需要修改源码，使用也非常简单。

**25. 共享模块文件夹 ► 引用的模块。**

共享模块功能在 B4X 早期非常有用。随着引用模块的引入，使用共享模块就没有什么意义了。引用模块涵盖了相同的用例，甚至更多。

**26. VideoView ► ExoPlayer**

ExoPlayer 功能更加强大，可定制性更强。

## 9 提示

这些是 Erel's 为 B4X 开发人员提供的建议（[\[B4X\] 为 B4X 开发人员提供的建议](#)）。

### 9.1 将数据与代码分离

将数据直接放入代码会使您的程序难以阅读且难以维护。

有很多简单的方法来处理数据。例如，您可以将文本文件添加到“文件”选项卡，然后使用以下命令将其读取到列表中：

```
Dim data As List = File.ReadList(File.DirAssets, "SomeFile.txt")
```

### 9.2 不要重复自己（DRY 原则）

如果您发现自己多次复制和粘贴相同的代码片段，然后做了一些小改动，那么最好停下来并尝试寻找更优雅的解决方案。

重复的代码很难维护和更新。Sender 关键字在许多情况下可以提供帮助（旧的但仍然相关的教程：[Tick-Tack-Toe: 使用视图数组](#)）。

### 9.3 地图集合

所有开发人员都应该知道如何使用 Map 集合。这是迄今为止最有用的集合。

教程：<https://www.b4x.com/android/forum/threads/map-collection-the-most-useful-collection.60304/>

### 9.4 新技术和新特点。

不要害怕学习新事物。作为开发人员，我们总是需要学习新事物。无论我们是否愿意，一切都在发展。我将以 [MQTT](#) 为例。我并不熟悉这项技术。当我开始学习它时，我惊讶地发现这个解决方案是多么简单和强大。

所有开发人员都应该知道的 B4X 特定功能：

- 智能字符串文字：<https://www.b4x.com/android/forum/threads/50135/#content>
- For Each 迭代器：<https://www.b4x.com/android/forum/threads/loops.57877/>
- 类：<https://www.b4x.com/android/forum/threads/18626/#content>

### 9.5 日志

您应该在应用程序运行时监控日志。特别是如果有任何错误。如果您由于某种原因无法看到日志，请花时间解决它。具体来说，使用 B4A-Bridge，日志只会在调试模式下出现。如果您遇到仅在发布模式下发生的问题，则需要切换到 USB 调试模式。

## 9.6 B4A 避免调用 DoEvents

DoEvents 会干扰内部消息队列。它可能会导致意外问题。只有极少数情况下需要它。B4A v1.0 发布时情况并非如此。从那时起，库已经发展起来，现在提供更好的解决方案。例如，如果数据库操作太慢（并且您正确使用了事务），那么您应该切换到异步方法。或者您应该使用 [Sleep](#) 或 [Wait For](#)。

## 9.7 字符串由字符而不是字节组成。

不要尝试将原始字节存储为字符串。这行不通。请改用字节数组。将字节转换为字符串的正确方法是使用 base 64 编码或 `ByteConverter.HexFromBytes`。

## 9.8 B4A 使用服务，尤其是入门服务

服务比活动简单。它们不会暂停，而且几乎总是可访问。

**关于全局变量的三个一般规则：**

1. 所有非 UI 相关变量都应在 `Process_Globals` 中声明。
2. 应在 Starter 服务的 `Service_Create` 中声明和设置/初始化公共（`process_global`）变量。
3. 仅当 `FirstTime` 为 `true` 时才应初始化活动进程全局变量。

这仅与 B4A 相关。B4J 和 B4i 更简单，因为没有特殊的生命周期，模块永远不会暂停。

## 9.9 UI 布局

B4X 提供了多种工具来帮助您实现适应所有屏幕尺寸的灵活布局。主要工具是：锚点和设计器脚本。避免添加多个变体（两个就可以了）。变体是在 v1.00 中引入的，早于其他功能。变体难以维护，可以用脚本替换。

锚点非常简单且功能强大。

不要过度使用百分比单位（除非您正在构建游戏）。

## 9.10 B4J 作为后端解决方案

B4A, B4i, B4J 共享相同的语言，相同的概念和大部分相同的 API。使用 `B4XSerializator` 在不同平台之间交换数据也很简单。

使用 B4J 可以轻松实现强大的服务器解决方案。特别是当客户端使用 B4A, B4i 或 B4J 实现时。

## 9.11 搜索

使用论坛搜索功能。您可以通过将平台（例如 B4A）添加到查询或单击结果页面中的某个过滤器来筛选结果。

论坛中提出的大多数问题都可以通过几次搜索来解决。

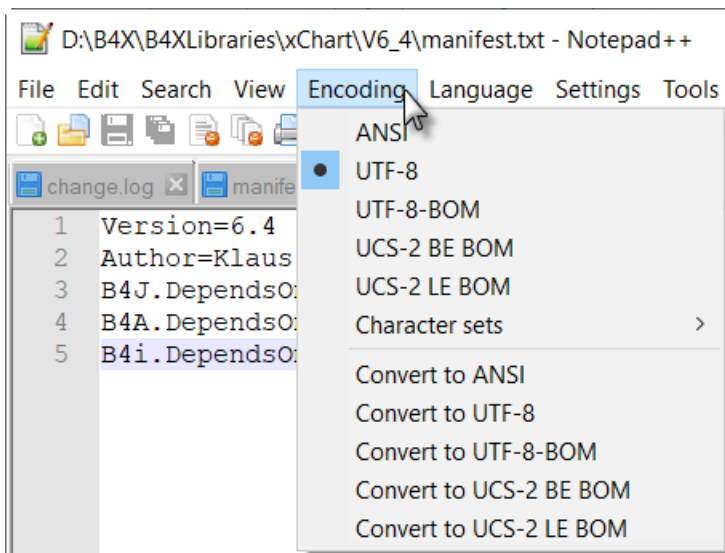


## 9.12 Notepad++

有时我们需要处理文本文件。我强烈建议所有开发人员使用一款优秀的文本编辑器，该编辑器可以显示编码、行尾字符和其他重要功能。<https://notepad-plus-plus.org/>

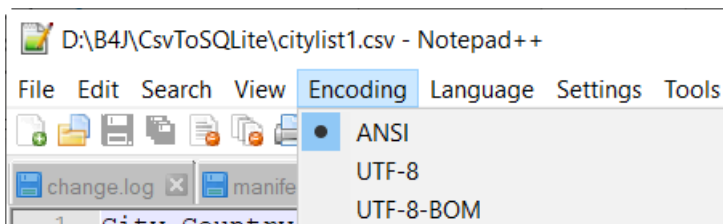
### 9.12.1 编码

要显示文本文件的当前编码，您可以加载它，然后在菜单中单击“编码”。当前编码已选中。您可以选择其他编码并保存文件。

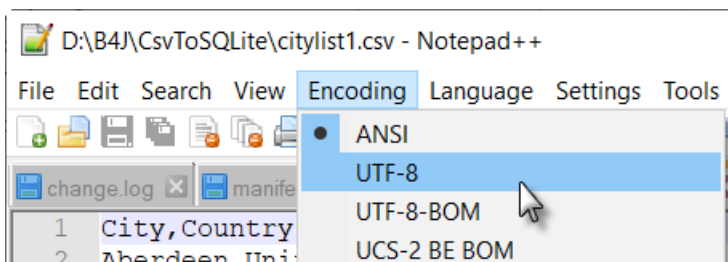


当您使用 Excel 生成的 csv 文件以 ANSI 编码进行编码，但 B4X 使用 UTF-8 编码时，这会很有用。

原始文件：



更改编码并使用另一个文件名保存文件。



当你重新加载此文件并检查编码时，你会看到以下内容：

