Livret B4x





B4x Langage Basic

Copyright: © 2017 Anywhere Software Edition 1.1

Dernière mise à jour : 2017.10.12

[1 BASIC 6](#_Toc495585714)

[1.1 B4x 6](#_Toc495585715)

[1.1.1 B4A Android 6](#_Toc495585716)

[1.1.2 B4i iOS 6](#_Toc495585717)

[1.1.3 B4J Java Desktop 7](#_Toc495585718)

[1.1.4 B4R Arduino 7](#_Toc495585719)

[2 Variables et objets 8](#_Toc495585720)

[2.1 Types de Variable 8](#_Toc495585721)

[2.2 Noms de variables 12](#_Toc495585722)

[2.3 Déclaration de variables 12](#_Toc495585723)

[2.3.1 Variables simples 12](#_Toc495585724)

[2.3.2 Tableaux (array) de variables 14](#_Toc495585725)

[2.3.3 Tableaux d’objets Views / Nodes 16](#_Toc495585726)

[2.3.4 Variables Type B4A, B4i et B4J seulement 19](#_Toc495585727)

[2.4 Assignation (Casting) 20](#_Toc495585728)

[2.5 Portée (Scope) 21](#_Toc495585729)

[2.5.1 Variables Processus 21](#_Toc495585730)

[2.5.2 Variables Activity B4A seulement 23](#_Toc495585731)

[2.5.3 Variables Locales 23](#_Toc495585732)

[2.6 Conseils 23](#_Toc495585733)

[3 Flux du programme / cycle de vie 24](#_Toc495585734)

[3.1 B4A 24](#_Toc495585735)

[3.1.1 Démarrage d’un projet 25](#_Toc495585736)

[3.1.2 Variables Process global 26](#_Toc495585737)

[3.1.3 Variables Activity 26](#_Toc495585738)

[3.1.4 Starter service 27](#_Toc495585739)

[3.1.5 Flux du programme 28](#_Toc495585740)

[3.1.6 Sub Process\_Globals / Sub Globals 29](#_Toc495585741)

[3.1.7 Sub Activity\_Create (FirstTime As Boolean) 29](#_Toc495585742)

[3.1.8 Résumé des déclarations de variables 30](#_Toc495585743)

[3.1.9 Sub Activity\_Resume Sub Activity\_Pause (UserClosed As Boolean) 31](#_Toc495585744)

[3.1.10 Activity.Finish / ExitApplication 32](#_Toc495585745)

[3.2 B4i 33](#_Toc495585746)

[3.3 B4J 34](#_Toc495585747)

[3.4 B4R 35](#_Toc495585748)

[3.5 Comparaison du flux B4A / B4i / B4J 36](#_Toc495585749)

[3.5.1 Démarrage du programme B4A / B4i / B4J 36](#_Toc495585750)

[3.5.2 Rotation du dispositif B4A / B4i 36](#_Toc495585751)

[4 Langage Basic 37](#_Toc495585752)

[4.1 Expressions 37](#_Toc495585753)

[4.1.1 Expressions mathématiques 37](#_Toc495585754)

[4.1.2 Expressions relationnelles 38](#_Toc495585755)

[4.1.3 Expressions booléennes 38](#_Toc495585756)

[4.2 Mots clé standard 39](#_Toc495585757)

[https://www.b4x.com/android/help/images/Method_636.pngAbs (Number As Double) As Double 41](#_Toc495585758)

[https://www.b4x.com/android/help/images/Method_636.pngACos (Value As Double) As Double 41](#_Toc495585759)

[https://www.b4x.com/android/help/images/Method_636.pngACosD (Value As Double) As Double 41](#_Toc495585760)

[https://www.b4x.com/android/help/images/Method_636.pngArray 41](#_Toc495585761)

[https://www.b4x.com/android/help/images/Method_636.pngAsc (Char As Char) As Int 41](#_Toc495585762)

[https://www.b4x.com/android/help/images/Method_636.pngASin (Value As Double) As Double 41](#_Toc495585763)

[https://www.b4x.com/android/help/images/Method_636.pngASinD (Value As Double) As Double 41](#_Toc495585764)

[https://www.b4x.com/android/help/images/Method_636.pngATan (Value As Double) As Double 41](#_Toc495585765)

[https://www.b4x.com/android/help/images/Method_636.pngATan2 (Y As Double, X As Double) As Double 41](#_Toc495585766)

[https://www.b4x.com/android/help/images/Method_636.pngATan2D (Y As Double, X As Double) As Double 41](#_Toc495585767)

[https://www.b4x.com/android/help/images/Method_636.pngATanD (Value As Double) As Double 41](#_Toc495585768)

[https://www.b4x.com/android/help/images/Method_636.pngBytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String 42](#_Toc495585769)

[https://www.b4x.com/android/help/images/Method_636.pngCallSub (Component As Object, Sub As String) As Object 42](#_Toc495585770)

[https://www.b4x.com/android/help/images/Method_636.pngCallSub2 (Component As Object, Sub As String, Argument As Object) As Object 42](#_Toc495585771)

[https://www.b4x.com/android/help/images/Method_636.pngCallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object 42](#_Toc495585772)

[https://www.b4x.com/android/help/images/Method_636.pngCatch 42](#_Toc495585773)

[https://www.b4x.com/android/help/images/Field.pngcE As Double 42](#_Toc495585774)

[https://www.b4x.com/android/help/images/Method_636.pngCeil (Number As Double) As Double 42](#_Toc495585775)

[https://www.b4x.com/android/help/images/Method_636.pngCharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String 43](#_Toc495585776)

[https://www.b4x.com/android/help/images/Method_636.pngChr (UnicodeValue As Int) As Char 43](#_Toc495585777)

[https://www.b4x.com/android/help/images/Method_636.pngContinue 43](#_Toc495585778)

[https://www.b4x.com/android/help/images/Method_636.pngCos (Radians As Double) As Double 43](#_Toc495585779)

[https://www.b4x.com/android/help/images/Method_636.pngCosD (Degrees As Double) As Double 43](#_Toc495585780)

[https://www.b4x.com/android/help/images/Field.pngcPI As Double 43](#_Toc495585781)

[https://www.b4x.com/android/help/images/Field.pngCRLF As String 43](#_Toc495585782)

[https://www.b4x.com/android/help/images/Method_636.pngDim 43](#_Toc495585783)

[https://www.b4x.com/android/help/images/Method_636.pngExit 43](#_Toc495585784)

[https://www.b4x.com/android/help/images/Field.pngFalse As Boolean 44](#_Toc495585785)

[https://www.b4x.com/android/help/images/Method_636.pngFloor (Number As Double) As Double 44](#_Toc495585786)

[https://www.b4x.com/android/help/images/Method_636.pngFor 44](#_Toc495585787)

[https://www.b4x.com/android/help/images/Method_636.pngGetType (object As Object) As String 44](#_Toc495585788)

[https://www.b4x.com/android/help/images/Method_636.pngIf 44](#_Toc495585789)

[https://www.b4x.com/android/help/images/Method_636.pngIsNumber (Text As String) As Boolean 44](#_Toc495585790)

[https://www.b4x.com/android/help/images/Method_636.pngLoadBitmap (Dir As String, FileName As String) As Bitmap 45](#_Toc495585791)

[https://www.b4x.com/android/help/images/Method_636.pngLoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap 45](#_Toc495585792)

[https://www.b4x.com/android/help/images/Method_636.pngLog (Message As String) 45](#_Toc495585793)

[https://www.b4x.com/android/help/images/Method_636.pngLogarithm (Number As Double, Base As Double) As Double 45](#_Toc495585794)

[https://www.b4x.com/android/help/images/Method_636.pngLogColor (Message As String, Color As Int) 45](#_Toc495585795)

[https://www.b4x.com/android/help/images/Method_636.pngMax (Number1 As Double, Number2 As Double) As Double 45](#_Toc495585796)

[https://www.b4x.com/android/help/images/Method_636.pngMe As Object 45](#_Toc495585797)

[https://www.b4x.com/android/help/images/Method_636.pngMin (Number1 As Double, Number2 As Double) As Double 45](#_Toc495585798)

[https://www.b4x.com/android/help/images/Method_636.pngNot (Value As Boolean) As Boolean 45](#_Toc495585799)

[https://www.b4x.com/android/help/images/Field.pngNull As Object 45](#_Toc495585800)

[https://www.b4x.com/android/help/images/Method_636.pngNumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String 46](#_Toc495585801)

[https://www.b4x.com/android/help/images/Method_636.pngNumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String 46](#_Toc495585802)

[https://www.b4x.com/android/help/images/Method_636.pngPower (Base As Double, Exponent As Double) As Double 46](#_Toc495585803)

[https://www.b4x.com/android/help/images/Field.pngQUOTE As String 46](#_Toc495585804)

[https://www.b4x.com/android/help/images/Field.pngRegex As Regex 46](#_Toc495585805)

[https://www.b4x.com/android/help/images/Method_636.pngReturn 46](#_Toc495585806)

[https://www.b4x.com/android/help/images/Method_636.pngRnd (Min As Int, Max As Int) As Int 46](#_Toc495585807)

[https://www.b4x.com/android/help/images/Method_636.pngRndSeed (Seed As Long) 46](#_Toc495585808)

[https://www.b4x.com/android/help/images/Method_636.pngRound (Number As Double) As Long 46](#_Toc495585809)

[https://www.b4x.com/android/help/images/Method_636.pngRound2 (Number As Double, DecimalPlaces As Int) As Double 46](#_Toc495585810)

[https://www.b4x.com/android/help/images/Method_636.pngSelect 47](#_Toc495585811)

[https://www.b4x.com/android/help/images/Method_636.pngSender As Object 47](#_Toc495585812)

[Sin (Radians As Double) As Double Calcule la fonction trigonométrique sinus. Angle en radians. 47](#_Toc495585813)



[SinD (Degrees As Double) As Double 47](#_Toc495585814)



[Calcule la fonction trigonométrique sinus. Angle en degrés. 47](#_Toc495585815)

[https://www.b4x.com/android/help/images/Method_636.pngSleep (Value As Double) As Double 47](#_Toc495585816)

[https://www.b4x.com/android/help/images/Method_636.pngSqrt (Value As Double) As Double 47](#_Toc495585817)

[https://www.b4x.com/android/help/images/Method_636.pngSub 48](#_Toc495585818)

[https://www.b4x.com/android/help/images/Method_636.pngSubExists (Object As Object, Sub As String) As Boolean 48](#_Toc495585819)

[https://www.b4x.com/android/help/images/Field.pngTAB As String 48](#_Toc495585820)

[https://www.b4x.com/android/help/images/Method_636.pngTan (Radians As Double) As Double 48](#_Toc495585821)

[https://www.b4x.com/android/help/images/Method_636.pngTanD (Degrees As Double) As Double 48](#_Toc495585822)

[https://www.b4x.com/android/help/images/Field.pngTrue As Boolean 48](#_Toc495585823)

[https://www.b4x.com/android/help/images/Method_636.pngTry 48](#_Toc495585824)

[https://www.b4x.com/android/help/images/Method_636.pngType 49](#_Toc495585825)

[https://www.b4x.com/android/help/images/Method_636.pngUntil 49](#_Toc495585826)

[https://www.b4x.com/android/help/images/Method_636.pngWhile 49](#_Toc495585827)

[4.3 Expressions conditionnelles 50](#_Toc495585828)

[4.3.1 If – Then – Elase 50](#_Toc495585829)

[4.3.2 Select – Case 52](#_Toc495585830)

[4.4 Structures de boucles (Loop) 54](#_Toc495585831)

[4.4.1 For – Next 54](#_Toc495585832)

[4.4.2 For - Each 55](#_Toc495585833)

[4.4.3 Do - Loop 56](#_Toc495585834)

[4.5 Subs / routines 58](#_Toc495585835)

[4.5.1 Déclaration 58](#_Toc495585836)

[4.5.2 Appel d’une Sub 58](#_Toc495585837)

[4.5.3 Appel d’une Sub depuis un autre module 58](#_Toc495585838)

[4.5.4 Noms 59](#_Toc495585839)

[4.5.5 Paramètres 59](#_Toc495585840)

[4.5.6 Valeur renvoyée 59](#_Toc495585841)

[4.6 Routines Sub ‘Resumable’ 60](#_Toc495585842)

[4.6.1 Sleep 60](#_Toc495585843)

[4.6.2 Wait For 61](#_Toc495585844)

[4.6.3 Flux du code 63](#_Toc495585845)

[4.6.4 Attendre qu’une routine ‘resumable sub’ se termine 64](#_Toc495585846)

[4.6.5 DoEvents 65](#_Toc495585847)

[4.6.6 Dialogues / Dialogs 66](#_Toc495585848)

[4.6.7 SQL avec Wait For 67](#_Toc495585849)

[4.6.7.1 Requêtes / Queries 67](#_Toc495585850)

[4.6.7.2 B4J 68](#_Toc495585851)

[4.6.8 Notes & Conseils 68](#_Toc495585852)

[4.7 Événements (Events) 69](#_Toc495585853)

[4.7.1 B4A 69](#_Toc495585854)

[4.7.2 B4i 72](#_Toc495585855)

[4.7.3 B4J 74](#_Toc495585856)

[4.7.4 B4R 77](#_Toc495585857)

[4.7.5 Résumé des objets d’interfaces utilisateur 78](#_Toc495585858)

[4.8 Bibliothèques (Libraries) 79](#_Toc495585859)

[4.8.1 Bibliothèques standard 80](#_Toc495585860)

[4.8.2 Dossier Bibliothèques additionnelles 81](#_Toc495585861)

[4.8.3 Téléchargement et mise à jour d’une bibliothèque 82](#_Toc495585862)

[4.8.4 Message d’erreur "Avez-vous oublié une référence de bibliothèque ?" 82](#_Toc495585863)

[4.9 Manipulations de textes / objet String 83](#_Toc495585864)

[4.9.1 B4A, B4i, B4J 83](#_Toc495585865)

[4.9.2 B4R 85](#_Toc495585866)

[4.10 Formatage de nombres 88](#_Toc495585867)

[4.10.1 B4A, B4i, B4J 88](#_Toc495585868)

[4.10.2 B4R 88](#_Toc495585869)

[4.11 Timers / Minuteurs 89](#_Toc495585870)

[4.12 Fichiers B4A, B4i, B4J 91](#_Toc495585871)

[4.12.1 Objet File (fichiers) 91](#_Toc495585872)

[4.12.2 Noms de fichiers 94](#_Toc495585873)

[4.12.3 Sous-dossiers 94](#_Toc495585874)

[4.12.4 B4A, B4J Objet TextWriter / écriture de textes 95](#_Toc495585875)

[4.12.5 B4A, B4J Objet TextReader / lecture de textes 96](#_Toc495585876)

[4.12.6 Encodage de texte 97](#_Toc495585877)

[4.13 Objet List / Liste B4A, B4i et B4J seulement 99](#_Toc495585878)

[4.14 Objet Map B4A, B4i et B4J seulement 101](#_Toc495585879)

[5 Outils d’aide 103](#_Toc495585880)

[5.1 Fonction recherche dans le forum / Search 103](#_Toc495585881)

[5.2 B4x Help Viewer 105](#_Toc495585882)

[5.3 Help documentation - B4A Object Browser 108](#_Toc495585883)

[6 Conseils 109](#_Toc495585884)

[6.1 Séparer les données du code 109](#_Toc495585885)

[6.2 Ne vous répétez pas 109](#_Toc495585886)

[6.3 Collection Map 109](#_Toc495585887)

[6.4 Nouvelles technologies et fonctionnalités 109](#_Toc495585888)

[6.5 Logs / affichage 109](#_Toc495585889)

[6.6 B4A, Evitez les appel de DoEvents 110](#_Toc495585890)

[6.7 L’objet Strings contient des caractères et non des octets 110](#_Toc495585891)

[6.8 B4A Utilisez des services, spécialement le service Starter 110](#_Toc495585892)

[6.9 Layouts IHM 110](#_Toc495585893)

[6.10 B4J une solution en arrière plan 110](#_Toc495585894)

[6.11 Recherche (Search) 111](#_Toc495585895)

[6.12 Notepad++. 111](#_Toc495585896)

[7 Dictionnaire 112](#_Toc495585897)

Contributeurs principaux : Klaus Christl (klaus), Erel Uziel (Erel)

Traduit par : Klaus Christl (klaus)

**Pour chercher un mot ou une phrase particulière veuillez utiliser la fonction Rechercher dans le menu Edition.**

Mis à jour pour les versions ci-dessous :

B4A version 7.30

B4i version 4.30

B4J version 5.90

B4R version 2.20

# BASIC

En programmation, le **BASIC** (acronyme pour Beginner's All-propose Symboliques Instruction Code, littéralement « Code d'instructions symboliques multi-usages du débutant »), est une famille de langages de programmation de haut niveau ayant pour but la facilité d'utilisation.

Le langage BASIC a été conçu en 1964 par John George Kemeny (1926-1993) et Thomas Eugene Kurtz (1928-) au « Dartmouth Collège » pour permettre aux étudiants des filières autre que scientifiques d'utiliser des ordinateurs. Les langages de programmation de l'époque étaient en effet plus adaptés à la carte perforée qu'aux terminaux interactifs. Même le Fortran II, peu évolué bien que déjà complexe, était inadapté aux calculs matriciels dont on faisait déjà usage en sciences humaines. Non interactifs, ces langages exigeaient qu'un programme ne comporte pas une seule erreur de syntaxe pour être exécuté (source Wikipedia).

B4R, B4A, B4J et B4i constituent ensemble la meilleure solution de développement pour l’**I**nternet **d**es **O**bjets (IdO), en anglais « Internet of Things » ou IoT)

## B4x

**B4X** est une suite d'éditeurs (IDE) de développement rapide d'applications qui permet la création d'applications pour les plates-formes suivantes: Google [Android](https://en.wikipedia.org/wiki/Android_%28operating_system%29), Apple [iOS](https://en.wikipedia.org/wiki/IOS), [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29), [Raspberry Pi](https://en.wikipedia.org/wiki/Raspberry_Pi) et [Arduino](https://en.wikipedia.org/wiki/Arduino). B4X utilise un dialecte propriétaire de Visual Basic (ci-après dénommé "B4X") et dispose également d'un concepteur visuel (Visual Designer) qui simplifie le processus de création d'interfaces utilisateur.

### B4A Android

B4A – La méthode simple pour développer des applications Android natives.

B4A comprend toutes les fonctionnalités nécessaires pour développer rapidement n'importe quel type d'application Android.  
B4A est utilisé par des dizaines de milliers de développeurs de partout dans le monde, y compris des entreprises telles que la NASA, HP, IBM et autres.  
Ensemble avec B4i, vous pouvez facilement développer des applications pour Android et iOS.

### B4i iOS

B4i – La méthode simple pour développer des applications iOS natives.

B4i est un outil de développement pour des applications iOS natives.  
B4i suit les mêmes concepts que B4A, vous permettant de réutiliser la plupart du code et de créer des applications pour Android et iOS.  
C'est le seul outil de développement qui vous permet de développer des applications iOS 100% natives sans ordinateur Mac local.

### B4J Java Desktop

B4J - Outil de développement moderne similaire à VisualBasic VB6 pour les solutions multi plate-forme de bureau, serveur et IOT.  
B4J est un outil de développement 100% gratuit.  
Avec B4J, vous pouvez facilement créer des applications de bureau (IHM, Interfac Homme Machine), des programmes de console (non-IHM) et des solutions de serveur.  
Les applications compilées peuvent être exécutées sur des cartes Windows, Mac, Linux et ARM (telles que Raspberry Pi).

### B4R Arduino

B4R - Créez facilement des programmes natifs Arduino et ESP8266.  
B4R est un outil de développement 100% gratuit.  
B4R suit les mêmes concepts que les autres outils B4X, fournissant un outil de développement simple et puissant.

# Variables et objets

Une **variable** est un nom symbolique donné à une quantité ou information connue ou inconnue, afin de permettre l'utilisation du nom indépendamment de l'information qu'il représente. Un nom de variable dans le code source d’un programme d'ordinateur est généralement associé à un emplacement de stockage de données et donc aussi à son contenu, et ceux-ci peuvent changer au cours de l'exécution du programme (source Wikipedia).

Il y a deux types de variables : primitives et non-primitives.

Primitives incluent les types numériques : Byte, Short, Int, Long, Float et Double.

Primitives incluent aussi : Boolean et Char.

## Types de Variable

**B4A, B4i, B4J**

Liste des types avec leur plages :

|  |  |  |  |
| --- | --- | --- | --- |
| B4x | Type | valeur min | valeur max |
| Boolean | booléen | False (faux) | True (vrai) |
| Byte | entier 8 bits | - 2 7 | 2 7 - 1 |
| -128 | 127 |
| Short | entier 16 bits | - 2 15 | 2 15 -1 |
| - 32768 | 32767 |
| Int | entier 32 bits | - 2 31 | 2 31 -1 |
| -2147483648 | 2147483647 |
| Long | entier long 64 bits | - 2 63 | 2 63 -1 |
| -9223372036854775808 | 9223372036854775807 |
| Float | nombre virgule flottante  32 bits | - 2 -149 | (2 -2 -23) \* 2 127 |
| 1.4E-45 | 3.4028235 E 38 |
| Double | nombre  double précision  64 bits | - 2 -1074 | (2 -2 -52) \* 2 1023 |
| 2.2250738585072014 E -308 | 1.7976931348623157 E 308 |
| Char | caractère |  |  |
| String | tableau de caractères |  |  |

**B4R**

Liste des types avec leur plages :

Types numériques :  
  
**Byte** 0 - 255  
**Int** (2 bytes) -32768 - 32768. Similaire au type Short type dans les autres langages B4x.  
**UInt** (2 bytes) 0 - 65535. Spécifique B4R.  
**Long** (4 bytes) -2,147,483,648 - 2,147,483,647. Similaire au type Int dans les autres B4x.  
**ULong** (4 bytes) 0 - 4,294,967,295 spécifique B4R.  
**Double** (4 bytes) virgule flottante 4 octets. Similaire à Float dans les autres langages B4x.  
Float est identique à Double. Short est identique à Int.  
  
Les indications ci-dessus sont valables pour toutes les cartes, y compris la Arduino Due.  
  
Autres types :  
**Boolean** True ou False (vrai ou faux). Sauvé en tant que octet (byte) avec les valeurs de 1 et 0.  
**String** Strings sont des tableaux de octets (bytes) finissant avec un octet nul (octet avec une valeur 0).  
**Object** Objets pouvant contenir n’importe quel type de valeurs.

Les types primitives sont toujours passés par valeur à d’autres routines (Sub) ou s’ils sont attribués à d’autres variables.

Par exemple :

Sub **S1**

Private A As Int

A = 12 La variable A = 12

S2(A) Elle est passée par valeur à la routine S2

Log(A) ' Affiche 12 La variable A est toujours égale à 12,   
 même que la variable B a été changée dans la routine S2.

End Sub

Sub **S2**(B As Int) Variable B = 12

B = 45 Sa valeur est changée à B = 45

End Sub

Tous les autres types, inclus des tableaux de types primitive ou des objets String sont considérés comme type non-primitif.

Si vous passez un type non-primitif à une routine ou si vous l’assignez à une autre variable, une copie de la référence est passée.

Ça veut dire que la donnée elle-même n’est pas dupliquée.

Ceci est différent de passer par référence car on ne peut pas changer la référence de la variable d’origine.

Tous les types peuvent être traités comme objets.

Les collections comme List et Map utilisent des objets et peuvent donc contenir n’importe quelle valeur.

Voici un exemple d’une erreur communément faite, où le développeur essaie d’ajouter plusieurs tableaux à un objet List :

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) 'Ajouter le tableau complet comme simple élément

Next

arr = List1.Get(0) 'Obtenir le premier élément de la List

Log(arr(0)) 'Qu’est-ce qui sera affiché ici???

Vous vous attendez à ce que 2 soit affiché. Toutefois c’est 10 qui sera affiché !  
Nous avons créé un tableau simple et ajouté 5 références de ce tableau à la List.  
Les valeurs dans le tableau sont les valeurs définies lors de la dernière itération.

Pour corriger ceci, nous devons créer un nouveau tableau à chaque itération.  
Ceci est fait en déclarant Private arr(3) dans chaque itération :

Private arr(3) As Int Cette ligne est redondante dans ce cas.

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) 'Ajouter le tableau complet comme simple élément

Next

arr = List1.Get(0) 'Obtenir le premier élément de la List

Log(arr(0)) 'Affichera 2

## Noms de variables

C'est à vous de définir le nom d’une variable, sauf pour les mots réservés.  
Un nom de variable doit commencer par une lettre et doit être composé des caractères suivants A-Z, a-z, 0-9 et souligné "\_", pas d'espaces, pas de parenthèses etc.  
Les noms de variables sont insensibles à la casse, cela signifie que *Indice* et *indice* se réfèrent à la même variable.

Mais, il est conseillé de leur donner des noms significatifs.

Exemple :

Interet = Capital \* Taux / 100 est significatif

n1 = n2 \* n3 / 100 pas significatif

Pour les objets Views (B4A, B4i), Nodes (B4J), il est utile d’ajouter au nom un préfixe de trois caractères qui définissent le type de l’objet.

Exemples :

lblCapital lbl > Label Capital > fonction

edtInterest edt > EditText Interet > fonction

btnSuivant btn > Button Suivant > fonction

## Déclaration de variables

### Variables simples

Les variables doivent être déclarées avec les mots clé Private ou Public suivis par le nom de la variable, puis le mot clé As et suivi par le type de variable. Pour plus de détails, voir [chapitre Portée](#_Scope).

Il existe aussi le mot clé Dim, qui est maintenu pour compatibilité.

Exemples :

Private Capital As Double Déclare trois variables en tant que *Double*,

Private Interet As Double nombres à double précision.

Private Taux As Double

Private i As Int Déclare deux variables en tant que *Int*, nombres entiers.

Private j As Int

Private lblCapital As Label Déclare deux variables en tant qu’objets Label.

Private lblInteret As Label

Private btnNext As Button Déclare deux variables en tant qu’objets Button.

Private btnPrev As Button

Les mêmes variables peuvent aussi être déclarées de manière plus courte.

Private Capital, Interet, Taux As Double

Private i, j As Int

Private lblCapital, lblInteret As Label

Private btnNext, btnPrev As Button

Les noms des variables séparées par des virgules et suivies par la déclaration du type.

Les déclarations ci-dessous sont aussi valables :

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

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

Les noms d’objets (View, Node) doivent être déclarés si on veut les utiliser dans le code.

Par exemple, si nous voulons changer le texte d’un objet Label dans le code tel que

lblCapital.Text = "1200",

nous devons au préalable référencer cet objet Label avec son nom lblCapital, ceci est réalisé avec la déclaration Private lblCapital As Label.

Si nous ne font jamais référence à cet objet Label n’importe où dans le code, aucune déclaration n’est nécessaire.

Si nous utilisons une routine événement pour cet objet, là aussi, aucune déclaration n’est nécessaire.

Pour allouer une valeur à une variable écrivez son nom suivi par le caractère égal et suivi par la valeur, comme :

Capital = 1200

NomDeFamille = "SMITH"

Notez que pour Capital nous avons écrit 1200 car Capital est un nombre.

Mais pour NomDeFamlle nous avons écrit "SMITH" car NomDeFamille est une variable texte (String).

Les textes doivent être mis entre deux guillemets.

### Tableaux (array) de variables

Les tableaux sont des collections de données ou d’objets qui peuvent être sélectionnés par des indices. Les tableaux peuvent avoir plusieurs dimensions.

Les déclarations comprennent le mot clé Private ou Public suivi par le nom de la variable NomDeFamille, le nombre d’éléments entre parenthèses (50), le mot clé As et le type de la   
variable String.

Pour des détails, voir le [chapitre Portée](#_Scope). Il existe aussi le mot clé Dim, qui a été maintenu pour compatibilité.

**Note : B4R ne supporte que des tableaux à une dimension !**

Exemples :

Public NomDeFamille(50) As String   
Tableau de ‘Strings’ à une dimension, nombre total d’éléments 50.

Public Matrix(3, 3) As Double   
Tableau de ‘Doubles’ à deux dimensions, nombre total d’éléments 9.

Public Data(3, 5, 10) As Int   
Tableau de ‘Int’ à trois dimensions, nombre total d’éléments 150.

Le premier index de chaque dimension dans un tableau est égal à 0.

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

Le dernier index est égal au nombre d’éléments dans chaque dimension moins 1.

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

Public PréNom (10) As String

Public NomDeFamille(10) As String

Public Adresse(10) As String

Public Ville(10) As String

ou

Public PréNom(10), NomDeFamille(10), Adresse(10), Ville(10) As String

L’exemple ci-dessous montre comment accéder aux éléments dans un tableau à trois dimensions.

Public Données(3, 5, 10) As Int

For i = 0 To 2

For j = 0 To 4

For k = 0 To 9

Données(i, j, k) = ...

Next

Next

Next

Une manière plus générale pour déclarer des tableaux est d’utiliser une variable.

Public NbPers = 10 As Int

Public PréNom(NbPers) As String

Public Nom(NbPers) As String

Public Adresse(NbPers) As String

Public Ville(NbPers) As String

Nous déclarons la variable Public NbPers = 10 As Int et lui attribuons la valeur 10.

Nous déclarons les tableaux avec cette variable au lieu du nombre 10 auparavant.

L’avantage de cette façon de déclarer des tableaux est lors’qu on doit une fois changer le nombre d’éléments on ne modifie qu’UNE seule valeur.

Pour le tableau Données, nous pourrions utiliser le code ci-dessous.

Public NbX = 2 As Int

Public NbY = 5 As Int

Public NbZ = 10 As Int

Public Données(NbX, NbY, NbZ) As Int

Et la routine d’accès.

For i = 0 To NbX - 1

For j = 0 To NbY - 1

For k = 0 To NbZ - 1

Données(i, j, k) = ...

Next

Next

Next

Remplir un tableau avec le mot clé Array :

Public Nom() As String

Nom = Array As String("Miller", "Smith", "Johnson", "Jordan")

### Tableaux d’objets Views / Nodes

Des objets Views / Nodes ou autres objets peuvent aussi être des tableaux (Array).

Dans B4A et B4i les objets d’interface utilisateur s’appellent *View* et s’appellent *Node* dans B4J.

Dans l’exemple ci-dessous les boutons (Buttons) sont ajoutés à l’objet View / Node parent dans le code.

**B4A**

Sub **Globals**

Private Boutons(6) As Button

End Sub

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

Private i As Int

For i = 0 To 5

Boutons (i).Initialize("Boutons")

Activity.AddView(Boutons(i), 10dip, 10dip + i \* 60dip, 150dip, 50dip)

Boutons (i).Tag = i + 1

Boutons (i).Text = "Test " & (i + 1)

Next

End Sub

Sub **Boutons\_Click**

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué")

End Sub

**B4i**

Sub **Process\_Globals**

Private Boutons(6) As Button

End Sub

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

Private i As Int

For i = 0 To 5

Boutons (i).Initialize("Boutons")

Page1.RootPanel.AddView(Boutons (i), 10dip, 10dip + i \* 60dip, 150dip, 50dip)

Boutons (i).Tag = i + 1

Boutons (i).Text = "Test " & (i + 1)

Next

End Sub

Sub **Boutons\_Click**

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué ")

End Sub

**B4J**

Sub **Process\_Globals**

Private Boutons(6) As Button

End Sub

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

Private i As Int

For i = 0 To 5

Boutons (i).Initialize("Boutons")

MainForm.RootPane.AddNode(Boutons (i), 10, 10 + i \* 60, 150, 50)

Boutons (i).Tag = i + 1

Boutons (i).Text = "Test " & (i + 1)

Next

End Sub

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

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué ")

End Sub

Les boutons (Buttons) auraient aussi pu être définis dans un fichier layout, dans ce cas, ils ne doivent pas être initialises, ni ajouté à un objet parent et les propriétés Text et Tag définies dans le concepteur visuel.

Dans ce cas, le code serait le suivant :

**B4A**

Sub **Globals**

Private b1, b2, b3, b4, b5, b6, b7 As Button

Private Boutons() As Button

End Sub

Sub Activity\_Create(FirstTime As Boolean)

Boutons = Array As Button(b1, b2, b3, b4, b5, b6, b7)

End Sub

Sub **Boutons\_Click**

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué ")

End Sub

**B4i**

Sub **Process\_Globals**

Private b1, b2, b3, b4, b5, b6, b7 As Button

Private Boutons(6) As Button

End Sub

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

Boutons = Array As Button(b1, b2, b3, b4, b5, b6, b7)

End Sub

Sub **Boutons\_Click**

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué ")

End Sub

**B4J**

Sub **Process\_Globals**

Private b1, b2, b3, b4, b5, b6, b7 As Button

Private Boutons(6) As Button

End Sub

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

Boutons = Array As Button(b1, b2, b3, b4, b5, b6, b7)

End Sub

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

Private btn As Button

btn = Sender

Log("Bouton " & btn.Tag & " cliqué")

End Sub

### Variables Type B4A, B4i et B4J seulement

**Une variable Type ne peut pas être privée (private). Une fois déclarée, elle est disponible n’importe où (similaire aux modules Class).**

L’endroit le plus approprié pour déclarer les Type est dans la routine Process\_Globals dans le module Main.

Reprenons l’exemple des données d’une personne.

Au lieu de déclarer chaque paramètre séparément, nous pouvons définir une variable personnalisée avec le mot clé Type :

Public NbUtilisateurs = 10 As Int

Type Personne(NomDeFamille As String, PréNom As String, Adresse As String, Ville As String)

Public Utilisateur(NbUtilisateurs) As Personne

Public UtilisateurActuel As Personne

Le nouveau type personnel est Personne, puis nous déclarons soit des variables uniques ou des tableaux de ce type.

Pour accéder à un élément particulier utilisez le code ci-dessous.

UtilisateurActuel.NomDeFamille

UtilisateurActuel.PréNom

Utilisateur(1).NomDeFamille

Utilisateur(1).PréNom

Le nom de la variable, suivi d’un point et le paramètre désiré.

Si la variable est un tableau, le nom est suivi de l’index désiré entre parenthèses.

On peut aussi assigner une variable type à une autre variable du même type comme ci-dessous.

UtilisateurActuel = Utilisateur(1)

## Assignation (Casting)

B4x assigne automatiquement les types aux besoins. Des nombres sont automatiquement convertis en objets String et vice et versa.

Dans beaucoup de cas vous devez manuellement assigner un type défini à un objet.

Cela se fait en assignant l’objet à une variable du type requis.

Par exemple, le mot clé Sender représente l’objet qui a généré un événement.

Le code ci-dessous change la couleur du bouton pressé.

Notez qu’il y a plusieurs boutons qui partagent la même routine d’événement.

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 ' Assigne l’objet à Button

btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))

End Sub

Le code ci-dessus peut être écrit de manière plus élégante :

Sub Globals

End Sub

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

Private i As Int

For i = 0 To 9 ' crée 10 Buttons

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 ' Cast the Object to Button

btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))

End Sub

## Portée (Scope)

### Variables Processus

Les variables sont actives tant que le processus est actif.

Vous devez déclarer ces variables dans la routine Sub Process\_Globals.

Cette routine est exécutée une seule fois lorsque le processus démarre (ceci est valable pour tous les modules, pas seulement pour le module principal Main).

Ces variables sont les seules variables “publiques (public)”. Ce qui veut dire qu’elles sont accessibles aussi depuis d’autres modules.

Néanmoins, dans B4A, pas tous les types de variables peuvent être déclarées en tant que variables processus.

Par exemple, des objets views / nodes ne peuvent pas être déclarées comme variables processus.

La raison est que nous ne voulons pas garder des références pour des objets qui sont détruits en même temps que l’Activity.

Autrement dit, lorsque l’Activity est détruite, tous les objets view qui sont contenus dans l’Activity seront aussi.

Si nous maintenions une référence pour une objet view, le collecteur de ressources (garbage collector) ne pourrait pas libérer les ressources et nous pourrions avoir un manque de mémoire. Le compilateur soutient cette exigence.

Pour accéder à des variables processus dans d’autres modules que celui où elles sont déclarées on doit ajouter à son nom le nom du module où elles sont déclarées en préfixe avec un point.

Exemple :

Variable définie dans un module avec le nom : *MonModule*

Sub **Process\_Globals**

Public MaVar As String

End Sub

Accéder à la variable dans le module *MonModule* :

MaVar = "Text"

Accéder à la variable dans un autre module :

MonModule.MaVar = "Text"

Les variables peuvent aussi être déclarées avec :

Dim MaVar As String

Dans ce cas elles sont considérées comme Public.

Mais, il est conseillé de les déclarer comme ci-dessous :

Public MaVar As String

Cette variable est publique.

On peut déclarer des variables privées dans la routine Sub Process\_Globals comme :

Private MaVar As String

Cette variable est privée pour le module dans lequel elle est déclarée.

Pour des Activities il est préférable de les déclarer dans la routine Sub Globals.

Pour des variables déclarées dans des modules Class dans la routine Sub Class\_Globals les mêmes règles que ci-dessus s’appliquent.

Public MyVarPublic As String ' publique

Private MyVarPublic As String ' privé

Dim MyVar As String ' publique identique à Public

**Utiliser le mot clé Dim dans Sub Class\_Globals n’est pas recommandé !**

### Variables Activity B4A seulement

Ces variables sont contenues dans l’Activity.

Vous devez déclarer ces variables dans la routine Sub Globals.

Ces variables sont "privées (private)" et ne sont accessibles que depuis le module de cette Activity.

Tous les types d’objets peuvent être déclarés en tant que variables d’Activity.

Chaque fois que l’Activity est créée, Sub Globals est exécuté avant Activity\_Create.

Ces variables existent seulement tant que l’Activity existe.

### Variables Locales

Les variables déclarées dans une sousroutine sont locales à cette sousroutine.

Elles sont "privées (private)" et ne sont disponibles que dans la routine où elles sont déclarées.

Tous les types d’objets peuvent être déclarés en tant que variables locales.

A chaque exécution de la routine, les variables locales sont initialisées à leur valeur par défaut ou à toute autre valeur que vous aurez définie dans le code et elles seront ‘détruites’ lors de la sortie de la routine.

## Conseils

Un objet view / node peut être assigné à une variable et vous pouvez aisément changer des propriétés communes de ces objets.

Par exemple, le code ci-dessous désactive tous les objets view / node appartenant à l’objet Panel / Pane :

For i = 0 To MonPanel.NumberOfViews - 1

Private v As View

v = MonPanel.GetView(i)

v.Enabled = False

Next

Si nous voulons désactiver que les boutons :

For i = 0 To MonPanel.NumberOfViews - 1

Private v As View

v = MonPanel.GetView(i)

If v Is Button Then ' vérifie si c’est un Button

v.Enabled = False

End If

Next

Note : MonPanel est un objet *Panel* dans B4A et B4i mais est un objet *Pane* dans B4J.

# Flux du programme / cycle de vie

## B4A

Commençons simplement :

Chaque programme B4A est execute dans son propre processus.

Un processus a une tâche principale (main thread) qui est aussi appellés tâche IU (interface utilisateur) (UI thread) qui est actif aussi longtemps que le processus. Un processus peut avoir plus qu’une tâche qui sont utiles pour des tâches en arrière-plan.

Un processus démarre lorsque l’utilisateur démarre votre programme, en admettant qu’il n’est pas exécuté en arrière-plan.

La fin du processus n’est pas vraiment déterminée. Cela survient quelque temps après que l’utilisateur ou le système n’aient ferme toutes les Activities.

Si, par exemple, vous avez une Activity et que l’utilisateur a pressé la touche Retour (back key), l’Activity sera fermée. Plus tard, quand le dispositive arrive à court de mémoire (ce qui peut arriver) le processus sera détruit.

Si l’utilisateur réexécute le programme et que le processus n’était pas détruit, alors le même processus sera repris.

Une application B4A est constituée d’une ou plusieurs Activities.

**Les Activities sont similaires aux fenêtres (forms) dans Windows.**

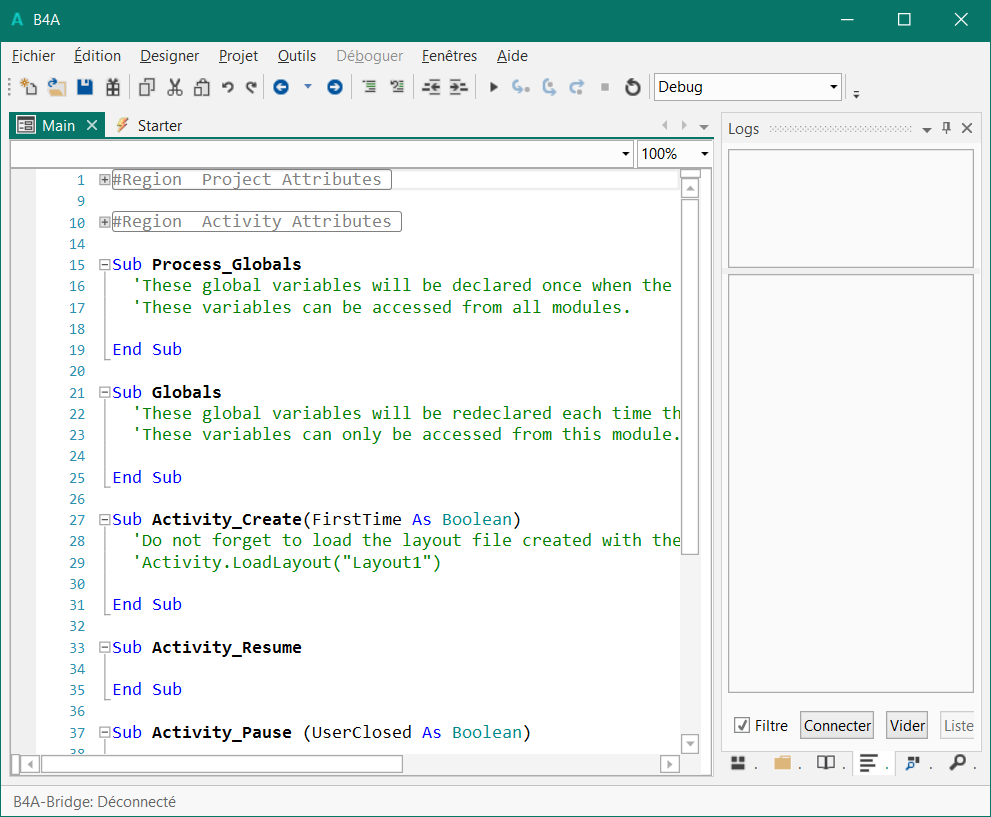
Une des différences majeures est que, quand une Activity n’est pas en avant-plan elle peut-être détruite pour preserver de la mémoire. Il est judicieu d’enregistrer l’état de l’Activity avant qu’elle ne soit perdue. Soit dans une mémoire persistante ou dans une mémoire qui est associée au processus.

Plus tard, cette Activity sera recréée en cas de besoin.

Un autre point délicat survient lorsqu’il y a un changement majeur de configuration. Le plus commun est un changement d’orientation (l’utilisateur tourne le dispositif). Lorsque un tel changement survient, les Activities actuelles sont détruites et recréées. Maintenant il est possible de recréer cette Activity par rapport à la nouvelle configuration (par exemple, nous connaissons les nouvelles dimensions de l’écran).

### Démarrage d’un projet

Lorsque nous démarrons un nouveau projet nous obtenons le modèle ci-dessous :



Sur le haut, nous voyons deux onglets module :

Main Activity

[Service Starter](#_Starter_service)

Le Service Starter est utilisé pour déclarer toutes les variables ProcessGlobal, et ces variables sont accessibles depuis n’importe quel module dans le projet.

L’Activity Main est l’Activity de démarrage et ne peut pas être supprimée.

Les variables peuvent être globales ou locales. Les variables locales sont des variables déclarées à l’intérieur d’une routine autre que Process\_Globals ou Globals.

Les variables locales sont locales dans la routine ou le module où elles sont déclarées. Lorsque la routine se termine, les variables n’existent plus.

Les variables globales sont accessibles depuis toutes les routines dans le module où elles sont déclarées.

Il y a deux types de variables globales.

Variables processus (accessible depuis tous les modules) et des variables ‘activity’ (accessibles que depuis un seul module).

### Variables Process global

Ces variables sont actives aussi longtemps que le processus est actif.

Vous devez déclarer ces variables comme Public dans la routine Sub Process\_Globals du Service Starter, comme ci-dessous.

Sub **Process\_Globals**

'These global variables will be declared once when the application starts.

'These variables can be accessed from all modules.

Public MaVariable = "Test" As String

Cette routine n’est exécutée qu’une seule fois au démarrage du processus.

Ces variables sont les seuls variables "publiques"., donc accessibles depuis tout autre module.

Il y a aussi une routine Process\_Globals dans chaque module Activity.

Si vous avez besoin de variables, valides uniquement dans cette Activity, qui ne sont initialisées qu’une seule fois au démarrage du programme vous pouvez les initialiser dans la routine Process\_Globals (ceci est valable pour toutes les Activities, pas seulement pour la première).

Néanmoins, pas tous les types d’objets peuvent être déclarés comme variables processus.

Toutes les Views ne peuvent pas être déclarées comme variables processus.

La raison est que nous ne pouvons pas garder des références à des objets qui seront détruits en même temps que l’Activity.

En d'autres termes, lorsque l'activité est détruite, toutes les Views contenues dans l'Activity sont également détruites. Si nous ne le faisions pas, et garderions des références pour des Views après que l’Activity ne soit détruite, le collecteur de ressources ne pourrait plus libérer ces ressources et pourrait conduire à un manque de mémoire.

Le compilateur soutient cette exigence.

### Variables Activity

Ces variables appartiennent à l’Activity.

Vous devez déclarer ces variables dans la routine Sub Globals.

Ces variables sont "privées" ("Private") et ne sont accessible que depuis le module Activity dans laquelle elles sont déclarées.

Tous les types d’objets peuvent être déclarées comme variables activity.

Chaque fois que l’Activity est créée, Sub Globals est exécuté (avant Activity\_Create).

Ces variables existent aussi longtemps que l’Activity existe.

### Starter service

Un des problèmes que les développeurs rencontrent avec de grandes applications est la multitude de points d’entrée possibles.  
  
Pendant le développement la majorité des applications démarrent avec l’Activity Main.  
Beaucoup d’applications démarrent avec du code similaire à celui-ci-dessous :

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

If FirstTime Then

SQL.Initialize(...)

SomeBitmap = LoadBitmap(...)

'code additionnel qui charge des resources pour toute l’application

End If

End Sub

Tout va bien pendant le développement. Mais, l’application plante de temps en temps sur le dispositif de l’utilisateur final.  
La raison de ces plantages est que le système d’exploitation peut démarrer le processus à partir d’une Activity ou d’un service différent. Par exemple, si vous utilisez StartServiceAt et que le système d’exploitation détruit le processus pendant qu’il est en arrière-plan.  
Dans ce cas, l’objet SQL et les autres ressources ne seront pas initialisées.  
  
A partir de B4A v5.20 il y a une nouvelle fonctionnalité appelée service Starter qui fournit un point d’entrée unique. Si le service Starter existe, le processus démarrera toujours à partir de ce service.  
  
Le service Starter sera créé et démarré, et seulement après l’Activity ou le service qui devait démarrer sera exécuté.  
Ce qui veut dire que le service Starter est le meilleur endroit pour initialiser toutes les ressources pour l’application.  
D'autres modules peuvent accéder en toute sécurité à ces ressources.  
Le service Starter devrait être l'emplacement par défaut pour toutes les variables publiques globales process. Des objets SQL, des données lues à partir de fichiers et des bitmaps utilisés par plusieurs Activities devraient être initialisées dans la routine Service\_Create dans le service Starter.  
  
Notes

* Le service Starter est identifié par son nom. Vous pouvez ajouter un nouveau service, avec le nom Starter à un projet existant et il deviendra le nouveau point d’entrée.  
  Ceci se fait dans le menu Projet > Ajouter un nouveau module > Module Service.
* C’est une fonctionnalité optionnelle. Vous pouvez supprimer le service Starter.
* Vous pouvez exécuter StopService(Me) dans Service\_Start si vous ne voulez pas que le service continue à fonctionner. Mais, ça signifie que le service ne sera plus à même de gérer des événements (vous ne pourrez plus, par exemple, utiliser les fonctions SQL asynchrones).
* Le service Starter doit être exclu de bibliothèques compilées.   
  Son attribut #ExcludeFromLibrary est True par défaut dans la Région Service Attributes.

### Flux du programme

Le flux du programme est le suivant :

* **Main Process\_Globals** routine Process\_Globals du module Main.   
  Ici nous déclarons toutes les variables et objets privés (Private) pour le module Main.
* **Starter Service Process\_Globals** Si le service existe la routine sera exécutée.  
  Ici nous déclarons toutes les variables et objets ‘Public Process Global’ tel que SQL, Bitmaps etc.
* **Autres Activity Main Process\_Globals** routines Process\_Globals d’autres modules.   
  Ici nous déclarons toutes les variables et objets privés (Private) pour le module concerné.
* **Starter Service Service\_Create** Si le service existe la routine sera exécutée.  
  Ici nous initialisons toutes les variables et objets ‘Public Process Global’ tel que SQL, Bitmaps etc.
* **Starter Service Service\_Start** Si le service existe la routine sera exécutée.  
  Nous laissons cette routine vide.
* [Globals](#_Globals_versus_FirstTime)  
  Ici nous déclarons toutes les variables et objets privés (Private) pour l’Activity concernée.
* [Sub Activity\_Create](#_Activity_Create_(FirstTime_As)  
  Ici nous chargeons les layouts et initialisons les objets Activity ajoutés dans le code.
* [Activity\_Resume](#_Sub_Activity_Resume_Sub)Cette routine est exécutée à chaque changement d’état de de l’Activity concernés.
* [Activity\_Pause](#_Sub_Activity_Resume_Sub)  
  Cette routine est exécutée lorsque l’Activity est mise en veille, changement d’orientation, lancement d’une autre Activtiy etc.

### Sub Process\_Globals / Sub Globals

Dans toute Activity, les routines Process\_Globals et Globals devraient être utilisées pour déclarer des variables.

Vous pouvez aussi attribuer des valeurs à des variables "simples" (nombres, strings et booléens).  
  
Vous ne devez y mettre aucun autre code.  
Vous devez mettre le code dans la routine Activity\_Create.

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

Cette routine est exécutée lorsque l’Activity est créée.

L’Activity est créée quand :

* l’utilisateur lance l’application la première fois.
* la configuration du dispositif a changé (rotation du dispositif) et que l’Activity a été détruite.
* l’Activity était en veille et que le système d’exploitation décide de la détruire pour libérer de la mémoire.

La fonction principale de cette routine est de charger ou créer les layouts (entre autres utilisations).

Le paramètre FirstTime (première fois) indique si c’est la première fois que l’Activtiy a été créée. First time se rapporte au processus actuel.

Vous pouvez utiliser FirstTime pour exécuter toutes sortes d’initialisations en rapport avec des variables processus.

Par exemple, si vous avez une liste avec des valeurs que vous devez lire, vous pouvez les lire quand FirstTime est True et les enregistrer dans une variable processus en déclarant la liste dans la routine Process\_Globals.

Maintenant nous savons que cette liste est disponible aussi longtemps que le processus est actif pas besoin de relire ces valeurs même si l’Activity est recréée.

En résumé, vous pouvez vérifier si FirstTime est True et initialiser les variables processus qui sont déclarées dans la routine Process\_Globals de l’Activity.

### Résumé des déclarations de variables

Quelle variable devons-nous déclarer où et où devons-nous initialiser cette variable :

* Les variables et objets non – interface homme machine que vous voulez utiliser dans plusieurs modules.  
  Comme SQL, Maps, Lists, Bitmaps etc.  
  Ces objets doivent être déclarés comme Public dans la routine Process\_Globals dans le service Starter comme :  
  Sub **Process\_Globals** Public SQL1 As SQL  
   Public Origin = 0 As Int  
   Public MyBitmap As Bitmap  
  End Sub  
    
  Et initialisées dans la routine Service\_Create dans le service Starter comme :  
  Sub **Service\_Create** SQL1.Initialize(...)  
   MyBitmap.Initialize(...)  
  End Sub
* Les variables accessibles depuis toutes les routines dans une Activity et qui ne sont initialisées qu’une seule fois.  
  Elles doivent être déclarées comme Private dans la routine Process\_Globals de l’Activity concernée comme :  
  Sub **Process\_Globals** Private MyList As List  
   Private MyMap As Map  
  End Sub  
    
  Et initialisées dans Activity\_Create comme :  
  Sub **Activity\_Create** MyList.Initialize  
   MyMap.Initialize  
  End Sub
* Variables dans un module Class ou Code.  
  Celles-ci sont généralement déclarées comme Private, vous pouvez les déclarer comme Public si vous voulez les utiliser à l’extérieur du module Class ou Code.   
  Les modules Class sont expliqués en détail dans le *B4x Booklet CustomViews* (en anglais).
* Objets interface homme-machine.  
  Ceux-ci doivent être déclarés dans Globals de l’Activity concernée comme :  
  Sub **Globals** Private btnGoToAct2, btnChangeValues As Button  
   Private lblCapital, lblInteret, lblTaux As Label  
  End Sub

Les variables simples tel que Int, Double, String ou Boolean peuvent être initialisées directement dans la ligne de déclaration même dans les routines Process\_Globals comme :

Public Origin = 0 as Int

**Aucun code ne doit être écrit dans les routines Process\_Globals !**

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

Activity\_Resume est exécuté juste après Activity\_Create ou après la reprise d’une Activity en veille (Activity mise en veille et reprise au premier plan).

Notez que lorsque vous ouvrez une Activity différente (en appelant StartActivity), l’Activity actuelle est d’abord mise en veille et l’autre Activity est créée si besoin mais toujours ‘resumed’ (exécution de Activity\_Resume).

Chaque fois que l’Activity est en veille (passant du premier plan vers l’arrière-plan), la routine Activity\_Pause est exécutée.

La routine Activity\_Pause est aussi exécutée quand l’Activtiy est active et qu’un changement de configuration survient, ce qui conduit à sa mise en veille puis à sa destruction.

Activity\_Pause est le dernier endroit pour enregistrer des informations importantes.

En général il y a deux mécanismes permettant d’enregistrer l’état d’une Activity. Des informations relevant seulement de l’instances courante de l’application peuvent être enregistrées dans une ou plusieurs variables processus.

D’autres informations devraient être enregistrées dans une mémoire persistante (fichier ou base de données).

Par exemple, si l’utilisateur a modifié quelques réglages, vous devez enregistrer ces changements dans une mémoire persistante dans cette routine. Sinon ces modifications peuvent être perdues.

La routine Activity\_Pause est exécutée chaque fois que l’Activity est désactivée (mise en veille). Ceci survient quand :

1. Une Activity différente est lancée.
2. Le bouton Home a été pressé.
3. Un événement de changement de configuration a été généré (changement d’orientation par exemple).
4. Le bouton Retour (Back) a été pressé.

Dans les cas 1 et 2, l’Activity sera mise en veille et maintenue en mémoire car elle est supposée être réutilisée plus tard.

Dans le cas 3, l’Activity sera mise en veille, détruite et créée et reprise (resumed) à nouveau.

Dans le cas 4, l’Activity sera mise en veille puis détruite. **Presser le bouton Retour (Back) est similaire à fermer l’Activity**. Dans ce cas il n’est pas nécessaire d’enregistrer des informations spécifiques à l’instance.

Le paramètre UserClosed est égal à True dans ce cas et False dans tous les autres. Notez qu’il sera aussi égal à True à l’exécution d’Activity.Finish. Cette fonction met aussi l’Activity en veille puis la détruit, similaire au bouton Retour (Back).

Vous pouvez utiliser le paramètre UserClosed pour définir quelles données seront enregistrées ou de réinitialiser des variables processus à leur valeur initiale.

### Activity.Finish / ExitApplication

Quelques explications sur comment et quand utiliser Activity.Finish ou ExitApplication.

Un article intéressant sur le fonctionnement d’Android peut être trouvé ici :

[Multitasking the Android way](http://android-developers.blogspot.com/2010/04/multitasking-android-way.html).

**En principe, on ne devrait pas utiliser ExitApplication mais lui préférer Activity.Finish qui laisse le système d’exploitation (OS) décider quand détruire le processus.**

**Vous ne devriez l’utiliser que si avez réellement besoin de détruire complètement le processus.**

Quand devrions-nous utiliser Activity.Finish et quand pas ?

Considérons un exemple sans Activity.Finish :

* **Main activity**
  + StartActivity(DeuxiemeActivity)
* **DeuxiemeActivity activity**
  + StartActivity(TroisiemeActivity)
* **TroisiemeActivity activity**
  + Clic sur le bouton Retour
  + L’OS retourne à l’Activity précédente, DeuxiemeActivity.
* **DeuxiemeActivity activity**
  + Clic sur le bouton Retour
  + L’OS retourne à l’Activity précédente, Main.
* **Main activity**
  + Clic sur le bouton Retour
  + L’OS quitte le programme

Considérons maintenant un exemple avec Activity.Finish avant chaque StartActivity :

* **Main activity**
  + Activity.Finish
  + StartActivity(DeuxiemeActivity)
* **DeuxiemeActivity activity**
  + Activity.Finish
  + StartActivity(TroisiemeActivity)
* **TroisiemeActivity activity**
  + Clic sur le bouton Retour
  + L’OS quitte le programme

Nous devrions utiliser Activity.Finish avant de démarrer une autre Activity seulement si nous ne voulons pas retourner vers cette Activity avec le bouton Retour.

## B4i

Le flux d’un programme est beaucoup plus simple dans B4i par rapport à B4A.

Lorsque vous lancez un nouveau projet vous obtenez le modèle de code ci-dessous :

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

Lorsque vous exécutez le programme, les routines sont exécutées dans l’ordre ci-dessus.

Notez que les dimensions de Page1 ne sont pas connues dans Application\_Start, elles ne sont connues que dans la routine Page1\_Resize dans les paramètres Width et Height.

Si vous voulez réajuster des objets, dimensions ou position, il faut le faire ici.

## B4J

Le flux d’un programme est beaucoup plus simple dans B4J par rapport à B4A, similaire à B4i.

Lorsque vous lancez un nouveau projet vous obtenez le modèle de code ci-dessous :

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") 'Load the layout file.

MainForm.Show

End Sub

'Return true to allow the default exceptions handler to handle the uncaught exception.

Sub **Application\_Error** (Error As Exception, StackTrace As String) As Boolean

Return True

End Sub

Lorsque vous exécutez le programme, les routines sont exécutées dans l’ordre ci-dessus.

Si vous voulez réajuster des objets lorsque l’utilisateur redimensionne une fenêtre (Form) vous devez ajouter une routine Resize pour cette fenêtre, comme :

Private Sub **MainForm\_Resize** (Width As Double, Height As Double)

' Votre code

End Sub

## B4R

Le flux d’un programme dans B4R est très simple.

Lorsque vous lancez un nouveau projet vous obtenez le modèle de code ci-dessous :

Sub **Process\_Globals**

'These global variables will be declared once when the application starts.

'Public variables can be accessed from all modules.

Public Serial1 As Serial

End Sub

Private Sub **AppStart**

Serial1.Initialize(115200)

Log("AppStart")

End Sub

Lorsque vous exécutez le programme, Process\_Globals puis AppStart sont exécutés.

Serial1.Initialize(115200) Initialise le débit binaire (bit rate).

Log("AppStart") Écrit “AppStart” dans l’onglet Logs.

## Comparaison du flux B4A / B4i / B4J

### Démarrage du programme B4A / B4i / B4J

**B4A B4i B4J**

Main Process\_Globals Main Process\_Globals Main Process\_Globals

Starter Process\_Globals

Autres modules Process\_Globals Autres modules Process\_Globals Autres 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

### Rotation du dispositif B4A / B4i

**B4A B4i**

Main Activity\_Pause

Main Globals Main Page1\_Resize

Main Activity\_Create

FirstTime = False

# Langage Basic

## Expressions

Dans les langages de programmation, une [expression](https://fr.wikipedia.org/wiki/Expression_(informatique)) est un élément de syntaxe qui combine un ensemble de lexèmes retournant une valeur.

C'est une combinaison de littéraux, de variables, d'opérateurs, et de fonctions qui est évaluée (ou calculée) en suivant les règles de priorité et d'associativité du langage de programmation pour produire (ou renvoyer) une nouvelle valeur (source Wikipedia).

Par exemple, 2 + 3 est une expression arithmétique qui vaut 5. Une variable est une expression car elle représente une valeur contenue en mémoire, donc y + 6 est une expression (source Wikipedia).

### Expressions mathématiques

|  |  |  |  |
| --- | --- | --- | --- |
| Operateur | Exemple | Niveau de priorité des opérations | Opération |
| + | x + y | 3 | Addition |
| - | x - y | 3 | Soustraction |
| \* | x \* y | 2 | Multiplication |
| / | x / y | 2 | Division |
| Mod | x Mod y | 2 | Modulo |
| Power | Power(x,y) xy | 1 | Puissance |

Niveau de priorité des opérations : Dans une expression, les opérations avec le niveau 1 sont exécutées avant les opérations de niveau 2, qui seront exécutées avant les opérations de niveau 3.

Exemples :

4 + 5 \* 3 + 2 = 21 > 4 + 15 + 2

(4 + 5) \* (3 + 2) = 45 > 9 \* 5

(4 + 5)2 \* (3 + 2) = 405 > 92 \* 5 > 81 \* 5

Power(4+5,2)\*(3+2)

11 Mod 4 = 3 > Mod est le reste de 11 / 4

233 Power(23,3) > 23 à la puissance 3

- 22 = - 4

(-2)2 = 4

### Expressions relationnelles

En informatique dans les expressions relationnelles, un opérateur teste une sorte de relation entre deux entités. Ceux-ci incluent une égalité numérique (par exemple, 5 = 5) et des inégalités (par exemple, 4> = 3).

Dans B4x, ces opérateurs renvoient **True** (vrai) ou **False** (faux), selon que la relation conditionnelle entre les deux opérandes est valable ou non.

|  |  |  |
| --- | --- | --- |
| Operateur | Exemple | Utilisé pour tester |
| = | x = y | L’équivalence de deux valeurs |
| <> | x <> y | Si les deux valeurs sont différentes |
| > | x > y | Si la valeur de gauche est plus grande que celle de droite |
| < | x < y | Si la valeur de gauche est plus petite que celle de droite |
| >= | x >= y | Si la valeur de gauche est plus grande ou égale à celle de droite |
| <= | x <= y | Si la valeur de gauche est plus petite ou égale à celle de droite |

### Expressions booléennes

Une expression booléenne est une expression qui donne une valeur booléenne (vrai ou faux). Par exemple, la valeur pour 5 > 3 est vrai et la valeur pour 5 < 4 est faux.

Les expressions booléennes sont utilisées pour le contrôle de flot et pour la sélection parmi un ensemble de valeurs selon un prédicat (source Wikipedia.

Des opérateurs booléens sont utilisés dans des expressions conditionnelles tel que IF-Then et

Select-Case.

|  |  |
| --- | --- |
| Operateur | Commentaire |
| Or | Ou Boolean Z = X Or Y Z = True si X ou Y est égal à True ou les deux sont True |
| And | Et Boolean Z = X And Y Z = True si X et Y sont les deux égal à True |
| Not ( ) | Non Boolean X = True Y = Not(X) > Y = False |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Or | And |
| X | Y | Z | Z |
| False | False | False | False |
| True | False | True | False |
| False | True | True | False |
| True | True | True | True |

## Mots clé standard

Certains mots clé ne sont pas disponible dans B4R.

https://www.b4x.com/android/help/images/Method_636.png  [**Abs**](#keywords_abs) (Number As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ACos**](#keywords_acos) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ACosD**](#keywords_acosd) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Array**](#keywords_array)  
https://www.b4x.com/android/help/images/Method_636.png  [**Asc**](#keywords_asc) (Char As Char) As Int  
https://www.b4x.com/android/help/images/Method_636.png  [**ASin**](#keywords_asin) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ASinD**](#keywords_asind) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ATan**](#keywords_atan) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ATan2**](#keywords_atan2) (Y As Double, X As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ATan2D**](#keywords_atan2d) (Y As Double, X As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**ATanD**](#keywords_atand) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**BytesToString**](#keywords_bytestostring) (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**CallSub**](#keywords_callsub) (Component As Object, Sub As String) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**CallSub2**](#keywords_callsub2) (Component As Object, Sub As String, Argument As Object) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**CallSub3**](#keywords_callsub3) (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**Catch**](#keywords_catch)  
https://www.b4x.com/android/help/images/Field.png  [**cE**](#keywords_ce) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Ceil**](#keywords_ceil) (Number As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**CharsToString**](#keywords_charstostring) (Chars() As Char, StartOffset As Int, Length As Int) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**Chr**](#keywords_chr) (UnicodeValue As Int) As Char  
https://www.b4x.com/android/help/images/Method_636.png  [**Continue**](#keywords_continue)  
https://www.b4x.com/android/help/images/Method_636.png  [**Cos**](#keywords_cos) (Radians As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**CosD**](#keywords_cosd) (Degrees As Double) As Double  
https://www.b4x.com/android/help/images/Field.png  [**cPI**](#keywords_cpi) As Double  
https://www.b4x.com/android/help/images/Field.png  [**CRLF**](#keywords_crlf) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**Dim**](#keywords_dim)  
https://www.b4x.com/android/help/images/Method_636.png  [**Exit**](#keywords_exit)  
https://www.b4x.com/android/help/images/Field.png  [**False**](#keywords_false) As Boolean  
https://www.b4x.com/android/help/images/Method_636.png  [**Floor**](#keywords_floor) (Number As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**For**](#keywords_for)  
https://www.b4x.com/android/help/images/Method_636.png  [**GetType**](#keywords_gettype) (object As Object) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**If**](#keywords_if)  
https://www.b4x.com/android/help/images/Method_636.png  [**Is**](#keywords_is)  
https://www.b4x.com/android/help/images/Method_636.png  [**IsNumber**](#keywords_isnumber) (Text As String) As Boolean  
https://www.b4x.com/android/help/images/Method_636.png  [**LoadBitmap**](#keywords_loadbitmap) (Dir As String, FileName As String) As Bitmap  
https://www.b4x.com/android/help/images/Method_636.png  [**LoadBitmapSample**](https://www.b4x.com/android/help/core.html#keywords_loadbitmapsample) (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap  
https://www.b4x.com/android/help/images/Method_636.png  [**Log**](#keywords_log) (Message As String)  
https://www.b4x.com/android/help/images/Method_636.png  [**Logarithm**](#keywords_logarithm) (Number As Double, Base As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**LogColor**](https://www.b4x.com/android/help/core.html#keywords_logcolor) (Message As String, Color As Int)  
https://www.b4x.com/android/help/images/Method_636.png  [**Max**](#keywords_max) (Number1 As Double, Number2 As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Me**](#keywords_me) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**Min**](#keywords_min) (Number1 As Double, Number2 As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Not**](#keywords_not) (Value As Boolean) As Boolean

[**Null**](#keywords_null) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**NumberFormat**](#keywords_numberformat) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**NumberFormat2**](#keywords_numberformat2) (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**Power**](#keywords_power) (Base As Double, Exponent As Double) As Double  
https://www.b4x.com/android/help/images/Field.png  [**QUOTE**](#keywords_quote) As String  
https://www.b4x.com/android/help/images/Field.png  [**Regex**](#keywords_regex) As Regex  
https://www.b4x.com/android/help/images/Method_636.png  [**Return**](#keywords_return)  
https://www.b4x.com/android/help/images/Method_636.png  [**Rnd**](#keywords_rnd) (Min As Int, Max As Int) As Int  
https://www.b4x.com/android/help/images/Method_636.png  [**RndSeed**](#keywords_rndseed) (Seed As Long)  
https://www.b4x.com/android/help/images/Method_636.png  [**Round**](#keywords_round) (Number As Double) As Long  
https://www.b4x.com/android/help/images/Method_636.png  [**Round2**](#keywords_round2) (Number As Double, DecimalPlaces As Int) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Select**](#keywords_select)  
https://www.b4x.com/android/help/images/Method_636.png  [**Sender**](#keywords_sender) As Object  
https://www.b4x.com/android/help/images/Method_636.png  [**Sin**](#keywords_sin) (Radians As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**SinD**](#keywords_sind) (Degrees As Double) As Double



https://www.b4x.com/android/help/images/Method_636.png  [**Sleep**](#keywords_sleep) (Milliseconds As Int)  
https://www.b4x.com/android/help/images/Method_636.png  [**SmartStringFormatter**](#keywords_smartstringformatter) (Format As String, Value As Object) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**Sqrt**](#keywords_sqrt) (Value As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**Sub**](#keywords_sub)  
https://www.b4x.com/android/help/images/Method_636.png  [**SubExists**](#keywords_subexists) (Object As Object, Sub As String) As Boolean  
https://www.b4x.com/android/help/images/Field.png  [**TAB**](#keywords_tab) As String  
https://www.b4x.com/android/help/images/Method_636.png  [**Tan**](#keywords_tan) (Radians As Double) As Double  
https://www.b4x.com/android/help/images/Method_636.png  [**TanD**](#keywords_tand) (Degrees As Double) As Double  
https://www.b4x.com/android/help/images/Field.png  [**True**](#keywords_true) As Boolean  
https://www.b4x.com/android/help/images/Method_636.png  [**Try**](#keywords_try)  
https://www.b4x.com/android/help/images/Method_636.png  [**Type**](#keywords_type)  
https://www.b4x.com/android/help/images/Method_636.png  [**Until**](#keywords_until)  
https://www.b4x.com/android/help/images/Method_636.png  [**While**](#keywords_while)

**https://www.b4x.com/android/help/images/Method_636.pngAbs (Number As Double) As Double**

Renvoie la valeur absolue de Number.

**https://www.b4x.com/android/help/images/Method_636.pngACos (Value As Double) As Double**

Calcule la fonction trigonométrique arccosinus. Renvoie l’angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngACosD (Value As Double) As Double**

Calcule la fonction trigonométrique arccosinus. Renvoie l’angle en degrés.

**https://www.b4x.com/android/help/images/Method_636.pngArray**

Crée un tableau à une dimension du type spécifié.  
Syntaxe: Array [As type] (liste de valeurs).  
Si le type est omis, un tableau d’objets sera créé.  
Exemple:   
Dim Jours() As String   
Jours = Array As String("Dimanche", "Lundi", ...)

**https://www.b4x.com/android/help/images/Method_636.pngAsc (Char As Char) As Int**

Renvoie le code unicode du caractère donné ou du premier caractère dans un objet String.

**https://www.b4x.com/android/help/images/Method_636.pngASin (Value As Double) As Double**

Calcule la fonction trigonométrique arcsinus. Renvoie l’angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngASinD (Value As Double) As Double**

Calcule la fonction trigonométrique arcsinus. Renvoie l’angle en degrés.

**https://www.b4x.com/android/help/images/Method_636.pngATan (Value As Double) As Double**

Calcule la fonction trigonométrique arctangente. Renvoie l’angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngATan2 (Y As Double, X As Double) As Double**

Calcule la fonction trigonométrique arctangente. Renvoie l’angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngATan2D (Y As Double, X As Double) As Double**

Calcule la fonction trigonométrique arctangente. Renvoie l’angle en degrés.

**https://www.b4x.com/android/help/images/Method_636.pngATanD (Value As Double) As Double**

Calcule la fonction trigonométrique arctangente. Renvoie l’angle en degrés.

**https://www.b4x.com/android/help/images/Method_636.pngBytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String**

Décode le tableau d’octets (bytes) en un objet String.  
Data - Le tableau d’octets.  
StartOffset - Le premier octet à lire.  
Length - Nombre d’octets à lire.  
CharSet - Le nom du codage de caractères.  
Exemple :   
Dim s As String   
s = BytesToString(Buffer, 0, Buffer.Length, "UTF-8")

**https://www.b4x.com/android/help/images/Method_636.pngCallSub (Component As Object, Sub As String) As Object**

Appel à la routine (Sub) donnée. CallSub peut être utilisé pour appeler une routine appartenant à un autre module.  
Néanmoins, la routine ne sera exécutée que si l’autre module est actif. Sinon, un objet String vide sera retourné.  
Vous pouvez utiliser IsPaused pour tester si un module est actif ou non.  
En B4A, cela veut dire que dans une Activity on ne peut pas appeler une routine dans une autre Activity car celle-ci est forcément inactive (Paused).  
Avec CallSub on peut appeler, depuis une Activity, une routine dans un service ou appeler depuis un service une routine dans une Activity.  
Notez qu’il n’est pas possible d’appeler des routines dans des modules Code.  
CallSub peut aussi être utilisé pour appeler une routine dans le même module. Passez Me en tant que Component dans ce cas.  
Exemple :   
CallSub(Main, "RefreshData")

**https://www.b4x.com/android/help/images/Method_636.pngCallSub2 (Component As Object, Sub As String, Argument As Object) As Object**

Similaire à CallSub. Appelle une routine avec un argument.

**https://www.b4x.com/android/help/images/Method_636.pngCallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object**

**https://www.b4x.com/android/help/images/Method_636.pngCatch**

Toute exception générée à l’intérieur d’un bloc Try sera captée dans le bloc Catch.  
Appelez LastException pour obtenir l’exception captée.  
Syntaxe :  
Try  
 ...  
Catch  
 Log(LastExeption)  
End Try

**https://www.b4x.com/android/help/images/Field.pngcE As Double**

Constate e (logarithme natural).

**https://www.b4x.com/android/help/images/Method_636.pngCeil (Number As Double) As Double**

Renvoie la plus petite valeur Double qui est plus grande ou égale à la valeur spécifiée mais égale à une valeur entière (integer).

**https://www.b4x.com/android/help/images/Method_636.pngCharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String**

Créé un nouvel objet String en copiant les caractères depuis le tableau.  
La copie commence depuis StartOffset et copie le nombre de caractères définis dans Length.

**https://www.b4x.com/android/help/images/Method_636.pngChr (UnicodeValue As Int) As Char**

Renvoie le caractère représenté par la valeur unicode donné.

**https://www.b4x.com/android/help/images/Method_636.pngContinue**

Arrête l’exécution de l’itération actuelle et continue avec la suivante.

**https://www.b4x.com/android/help/images/Method_636.pngCos (Radians As Double) As Double**

Calcule la fonction trigonométrique cosinus. Angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngCosD (Degrees As Double) As Double**

Calcule la fonction trigonométrique cosinus. Angle en degrés.

**https://www.b4x.com/android/help/images/Field.pngcPI As Double**

Constante PI.

**https://www.b4x.com/android/help/images/Field.pngCRLF As String**

Constante saut de ligne. Valeur de Chr(10).

**https://www.b4x.com/android/help/images/Method_636.pngDim**

Déclare une variable. Obsolète, utilisez Private ou Public à la place.  
Syntaxe :  
Déclarer une variable simple :  
Dim nom variable [As type] [= expression]  
Le type par défaut est String.  
  
Déclarer des variables multiples. Toutes les variables seront du type spécifié.  
Dim [Const] variable1 [= expression], variable2 [= expression], ..., [As type]  
Notez que la syntaxe courte ne s’applique qu’au mot clé Dim.  
Exemple : Dim a = 1, b = 2, c = 3 As Int  
  
Déclarer un tableurs :  
Dim variable(Rang1, Rang2, ...) [As type]  
Exemple: Dim Days(7) As String  
On peut omettre le rang pour un tableau de longueur zéro.

**https://www.b4x.com/android/help/images/Method_636.pngExit**

Sort de la boucle le plus intérieure.  
Notez que Exit dans un bloc Select sort de ce bloc Select.

**https://www.b4x.com/android/help/images/Field.pngFalse As Boolean**

**https://www.b4x.com/android/help/images/Method_636.pngFloor (Number As Double) As Double**

Renvoie la plus grande valeur Double qui est plus petite ou égale à la valeur spécifiée mais égale à une valeur entière (integer).

**https://www.b4x.com/android/help/images/Method_636.pngFor**

Syntaxe:  
For variable = valeur1 To valeur2 [Step intervalle]  
 ...  
Next  
Si le type de l’itérateur n’a pas été spécifié auparavant il sera du type Int.  
  
Ou:  
For Each variable As type In collection  
 ...  
Next  
Exemples :   
For i = 1 To 10   
 Log(i) 'Affiche 1 à 10 (inclusif).   
Next   
For Each n As Int In Numbers 'un tableau   
 Sum = Sum + n   
Next   
  
Notez que la limite de la boucle n’est calculée qu’une fois avant la première itération.

**https://www.b4x.com/android/help/images/Method_636.pngGetType (object As Object) As String**

Renvoie un objet String représentant le type Java de l’objet.

**https://www.b4x.com/android/help/images/Method_636.pngIf**

Sur une ligne:  
If condition Then expression-vraie [Else expression-fausse]  
Plusieurs lignes :  
If condition Then  
 expression  
Else If condition Then  
 expression  
 ...  
Else  
 expression  
End If

**https://www.b4x.com/android/help/images/Method_636.pngIsNumber (Text As String) As Boolean**

Teste si l’objet String peut être converti en nombre.

**https://www.b4x.com/android/help/images/Method_636.pngLoadBitmap (Dir As String, FileName As String) As Bitmap**

Charge une bitmap.  
Notez que le système de fichier dans Android est sensible à la casse.  
Prenez en considération l’utilisation de LoadBitmapSample si la taille de l’image est grande.  
La taille du fichier de l’image ne correspond pas à la taille de l’image car le fichier est compressé.  
Exemple:   
Activity.SetBackgroundImage(LoadBitmap(File.DirAssets, "UnFichier.jpg"))

**https://www.b4x.com/android/help/images/Method_636.pngLoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap**

Charge une bitmap.  
Le décodeur va sous-échantillonner la bitmap si MaxWidth (largeur max) ou MaxHeight (hauteur max) sont plus petits que les dimensions de la bitmap.  
Ceci peut économiser beaucoup de mémoire si on charge de grandes images.  
Exemple:   
Panel1.SetBackgroundImage(LoadBitmapSample(File.DirAssets, "UnFichier.jpg", Panel1.Width, Panel1.Height))

**https://www.b4x.com/android/help/images/Method_636.pngLog (Message As String)**

Affiche (Logs) un message. Le message est affiché dans l’onglet Logs.

**https://www.b4x.com/android/help/images/Method_636.pngLogarithm (Number As Double, Base As Double) As Double**

**https://www.b4x.com/android/help/images/Method_636.pngLogColor (Message As String, Color As Int)**

Affiche (Logs) un message. Le message sera affiché, dans l’onglet Logs, avec la couleuse spécifiée.

**https://www.b4x.com/android/help/images/Method_636.pngMax (Number1 As Double, Number2 As Double) As Double**

Renvoie la plus grande des deux valeurs.

**https://www.b4x.com/android/help/images/Method_636.pngMe As Object**

Pour les classes : renvoie une référence à l’instance actuelle.  
Pour les Activities et Services : renvoie une référence à un objet qui peut être utilisé avec les mots clé CallSub, CallSubDelayed et SubExists.  
Ne peut pas être utilisé dans des modules Code.

**https://www.b4x.com/android/help/images/Method_636.pngMin (Number1 As Double, Number2 As Double) As Double**

Renvoie la plus petite des deux valeurs.

**https://www.b4x.com/android/help/images/Method_636.pngNot (Value As Boolean) As Boolean**

Inverse la valeur booléenne donnée.

**https://www.b4x.com/android/help/images/Field.pngNull As Object**

**https://www.b4x.com/android/help/images/Method_636.pngNumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String**

Convertit le nombre spécifié en un objet String.   
L’objet String contiendra au moins MinimumIntegers chiffres avant le point décimal et au plus MaximumFractions chiffres après le point décimal.  
Exemple:   
Log(NumberFormat(12345.6789, 0, 2)) '"12,345.68"   
Log(NumberFormat(1, 3 ,0)) '"001"

**https://www.b4x.com/android/help/images/Method_636.pngNumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String**

Convertit le nombre spécifié en un objet String.

L’objet String contiendra au moins MinimumIntegers chiffres avant le point décimal et au plus MaximumFractions et au moins MinimumFractions chiffres après le point décimal.  
GroupingUsed – Détermine si le caractère de groupage est affiché ou non.  
Exemple :   
Log(NumberFormat2(123456789.12, 0, 3, 3, False)) '"123456789.120"

Log(NumberFormat2(123456789.12, 0, 3, 3, True)) '"12’3456’789.120"

**https://www.b4x.com/android/help/images/Method_636.pngPower (Base As Double, Exponent As Double) As Double**

Renvoie la valeur de Base élevé à la puissance Exponent.

**https://www.b4x.com/android/help/images/Field.pngQUOTE As String**

Caractère guillemet ". La valeur de Chr(34).

**https://www.b4x.com/android/help/images/Field.pngRegex As Regex**

Méthodes d’[expressions régulières](https://fr.wikipedia.org/wiki/Expression_r%C3%A9guli%C3%A8re).

**https://www.b4x.com/android/help/images/Method_636.pngReturn**

Renvoie depuis la routine (Sub) courante et renvoie optionnellement la valeur spécifiée.  
Syntaxe : Return [value]

**https://www.b4x.com/android/help/images/Method_636.pngRnd (Min As Int, Max As Int) As Int**

Renvoie une nombre entier aléatoire compris entre Min (inclusif) et Max (exclusif).

**https://www.b4x.com/android/help/images/Method_636.pngRndSeed (Seed As Long)**

Définit la valeur de départ (Seed) pour le nombre aléatoire.   
Cette fonction sert lors du débogage pour toujours obtenir le même résultat.

**https://www.b4x.com/android/help/images/Method_636.pngRound (Number As Double) As Long**

Renvoie le nombre donné arrondi au nombre entier (Long) le plus proche.

**https://www.b4x.com/android/help/images/Method_636.pngRound2 (Number As Double, DecimalPlaces As Int) As Double**

Renvoie le nombre donné arrondi au nombre de décimales données.

**https://www.b4x.com/android/help/images/Method_636.pngSelect**

Compare une valeur simple à des valeurs multiples.  
Exemple:   
Dim valeur As Int   
valeur = 7   
Select valeur   
  Case 1   
    Log("Un")   
  Case 2, 4, 6, 8   
    Log("Pair")   
  Case 3, 5, 7, 9   
    Log("Impair plus grand que un")   
  Case Else   
    Log("Plus grand que 9")   
End Select

**https://www.b4x.com/android/help/images/Method_636.pngSender As Object**

Renvoie l’objet qui a généré l’événement (event).  
Valable uniquement dans une routine événement.  
Exemple:   
Sub **Button\_Click**   
 Dim b As Button   
 b = Sender   
 b.Text = "J’ai été cliqué"   
End Sub

**Sin (Radians As Double) As Double**Calcule la fonction trigonométrique sinus. Angle en radians.



**SinD (Degrees As Double) As Double**



Calcule la fonction trigonométrique sinus. Angle en degrés.

**https://www.b4x.com/android/help/images/Method_636.pngSleep (Value As Double) As Double**

Arrête l’exécution de la routine courante, puis continue après expiration du temps écoulé.

**https://www.b4x.com/android/help/images/Method_636.pngSmartStringFormatter (Format As String, Value As Object) As String**

Mot clé interne utilisé par la fonction Smart String.

**https://www.b4x.com/android/help/images/Method_636.pngSqrt (Value As Double) As Double**

Renvoie la racine carrée positive.

**https://www.b4x.com/android/help/images/Method_636.pngSub**

Déclare une routine avec les paramètres et le type d’objet à renvoyer.  
Syntaxe: nom de la Sub [(liste de paramètres)] [As type de retour]  
Les paramètres incluent leur nom et leur type.  
La longueur de tableaux ne doit pas être donnée.  
Exemple :   
Sub **MaSub** (Nom As String, PréNom As String, Age As Int, AutresValeurs() As Double) As Boolean   
 ...   
End Sub  
Dans cet exemple AutresValeurs est un tableau à une dimension.  
La déclaration du type de retour est différente que dans les autres déclarations, les parenthèses suivent le type et non le nom (ce qui n’est pas le cas dans l’exemple).

**https://www.b4x.com/android/help/images/Method_636.pngSubExists (Object As Object, Sub As String) As Boolean**

Teste si l’objet inclut la routine spécifiée on non.  
Renvoie False si l’objet n’est pas initialisé ou qu’il n’est pas une instance d’une classe utilisateur.

**https://www.b4x.com/android/help/images/Field.pngTAB As String**

Caractère de tabulation Tab.

**https://www.b4x.com/android/help/images/Method_636.pngTan (Radians As Double) As Double**

Calcule la fonction trigonométrique tangente. Angle en radians.

**https://www.b4x.com/android/help/images/Method_636.pngTanD (Degrees As Double) As Double**

Calcule la fonction trigonométrique tangente. Angle en degrés.

**https://www.b4x.com/android/help/images/Field.pngTrue As Boolean**

**https://www.b4x.com/android/help/images/Method_636.pngTry**

Toute exception générée à l’intérieur d’un bloc Try sera captée dans le bloc Catch.  
Appelez LastException pour obtenir l’exception captée.  
Syntaxe :

Try  
...  
Catch  
...  
End Try

**https://www.b4x.com/android/help/images/Method_636.pngType**

Déclare une structure.  
Ne peut être utilisé que dans Sub Globals ou Sub Process\_Globals.  
Syntaxe :  
Type nom-type (Champ1, Champ2, ...)  
Champ inclut nom et type.  
Exemple:   
Type MonType (Nom As String, Elément(10) As Int)   
Dim a, b As MonType   
a.Initialize   
a.Elément(2) = 123

**https://www.b4x.com/android/help/images/Method_636.pngUntil**

Passe dans la boucle jusqu’à ce que la condition est vérifiée (égale à True).  
Syntaxe :  
Do Until condition  
 ...  
Loop

**https://www.b4x.com/android/help/images/Method_636.pngWhile**

Passe dans la boucle tant que la condition est vérifiée (égale à True).  
Syntaxe :  
Do While condition  
...  
Loop

## Expressions conditionnelles

Différentes exécutions conditionnelles sont disponibles dans Basic.

### If – Then – Elase

La structure **If-Then-Else** (si-alors-ou bien) permet d’effectuer des tests conditionnels et exécuter différentes sections de code en fonction des résultats des tests.

General case :

If test1 Then

' code1

Else If test2 Then

' code2

Else

' code3

End If

La structure **If-Then-Else** fonctionne comme suit :

1. Lorsqu’on atteint la ligne avec le mot clé **If**, **test1** est effectué.
2. Si le résultat est **True**, alors **code1** est exécuté jusqu’avant la ligne contenant les mots clé **Else If**. Puis, saute à la ligne juste après celle contenant les mots clé **End If** et continue.
3. Si le résultat est **False**, alors **test2** est effectué.
4. Si le résultat est **True**, alors **code2** est exécuté jusqu’avant la ligne contenant le mot clé **Else**. Puis, saute à la ligne juste après celle contenant les mots clé **End If** et continue.
5. Si le résultat est **False**, alors **code3** est exécuté puis continue à la ligne juste après celle contenant les mots clé **End If**.

Ces tests peuvent être n’importe quel test conditionnel avec deux réponses possibles **True** ou **False**.

Quelques exemples :

If b = 0 Then

a = 0 La structure **If-Then** la plus simple.

End If

If b = 0 Then a = 0 La même, mais sur une seule ligne.

If b = 0 Then

a = 0 La structure **If-Then-Else** la plus simple.

Else

a = 1

End If

If b = 0 Then a = 0 Else a = 1 La même, mais sur une seule ligne.

Personnellement, je préfère la structure sur plusieurs lignes, meilleure lisibilité.

Vieille habitude du Basic HP d’il y a quelques dizaines d’années, ce Basic n’autorisait qu’une seule instruction par ligne.

Note. Différence entre :

B4x VB

**Else If** **ElseIf**

Dans B4x il y a un blanc entre les deux mots **Else** et **If**, alors que c’est en un mot dans VB

Quelques utilisateurs utilisent cette structure :

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

Il y a une grande différence entre B4x et VB qui entraîne des erreurs :

L’instruction ci-dessus est équivalente à :

B4x VB

If b = 0 Then If b = 0 Then

a = 0 a = 0

End If c = 1

c = 1 End If

Le caractère double point ' : ' dans l’instruction est traité dans B4x comme un caractère CR retour chariot.

Cette structure génère une erreur.  
Sub Plus1 : x = x + 1 : End Sub

On ne peut pas avoir la déclaration Sub d’une routine et End Sub sur une même ligne.

### Select – Case

La structure **Select - Case** (Sélection **-** Cas) permet de comparer une **ExpressionTest** avec d’autres **Expressions** et d’exécuter différentes sections de code en fonction des correspondances entre **ExpressionTest** et **Expressions**.

General case :

Select ExpressionTest **ExpressionTest** est l’expression à tester.

Case ListeExpression1

' code1 **ListeExpression1** est une liste d’expressions à comparer

Case ListeExpression 2 avec **ExpressionTest**

' code2 **ListeExpression2** est une autre liste d’expressions à

Case Else compare avec **ExpressionTest**

' code3

End Select

La structure **Select - Case** fonctionne comme suit :

1. Le test **ExpressionTest** est effectué.
2. Si un des éléments dans **ListeExpression1** correspond à **ExpressionTest** , alors **code1** est exécuté puis le programme continue depuis la ligne juste après le mot clé **End Select**.
3. Si un des éléments dans **ListeExpression2** correspond à **ExpressionTest,** alors **code2** est exécuté puis le programme continue depuis la ligne juste après le mot clé **End Select**.
4. Si aucune expression ne correspond à **TestExpression**, alors **code3**   
   est exécuté puis le programme continue depuis la ligne juste après le mot clé **End Select**.

**ExpressionTest** peut être n’importe quelle expression ou une valeur.

**ListeExpression1** est une liste de n’importe quelle expression ou valeurs.

Exemples :

Select Valeur

Case 1, 2, 3, 4 La variable Valeur est numérique.

Select a + b L’expression test **ExpressionTest** est la somme de a + b

Case 12, 24

Select Txt.CharAt L’expression test **ExpressionTest** est un caractère.

Case "A", "B", "C"

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

Note. Différences entre :

B4x VB

Select Value Select Case Value

Case 1,2,3,4,8,9,10 Case 1 To 4 , 8 To 9

Dans VB le mot clé Case est ajouté après le mot clé Select.

VB accepte Case 1 To 4 , ceci n’est pas implémenté dans B4x.

## Structures de boucles (Loop)

Différentes structures de boucle sont disponibles dans B4x.

### For – Next

Dans une boucle **For–Next**, le même code sera exécuté un certain nombre de fois.

Exemple :

For i = n1 To n2 Step n3 i variable d’incrément

n1 valeur initiale

' Specific code n2 valeur finale

n3 pas

Next

Une boucle **For–Next** fonctionne comme suit :

1. Au début, la variable d’incrément **i** est égale à la valeur initiale **n1**.  
   i = n1
2. Le code entre les mots clé **For** et **Next** est exécuté.
3. Lorsqu’on atteint **Next**, la variable d’incrément **i** est incrémentée de la valeur ‘Step’ **n3**.  
   i = i + n3.
4. Le programme revient vers **For**, compare si la valeur d’incrément **i** est inférieure à la valeur finale **n2**.  
   test si i <= n2
5. Si **Oui**, le programme continue au point 2, la ligne juste après le mot clé **For**.
6. Si **Non**, le programme continue à la ligne juste après le mot clé **Next**.

Si la valeur du pas (Step) est égale à '+1', le mot clé **Step** et sa valeur ne sont pas nécessaire.

For i = 0 To 10 For i = 0 To 10 Step 1

est identique à

Next Next

La valeur de la variable pas (Step) peut être négative.

For i = n3 To 0 Step -1

Next

On peut sortir d’une boucle For - Next avec le mot clé **Exit**.

For i = 0 To 10 Dans l’exemple, si la valeur A est égale à 0.

' code

If A = 0 Then Exit Alors on quitte la boucle.

' code

Next

**Note :** Différences entre

B4x VB

Next Next i

Exit Exit For

Dans VB :

* La variable d’incrément est ajoutée après le mot clé **Next**.
* Le type de boucle, **For** dans l’exemple, est ajouté après le mot clé **Exit**.

### For - Each

C’est une variante de la boucle For - Next.

Exemple :

For Each n As Type In Array n variable, n’importe quel objet

Type type de la variable n

' Specific code Array Tableau (Array) de variables objets

Next

La boucle **For–Each** fonctionne comme suit :

1. Au début, **n** reçoit la valeur du premier élément dans le tableau (Array).  
   n = Array(0)
2. Le code entre les mots clé **For** et **Next** est exécuté.
3. Lorsqu’on atteint **Next**, le programme teste si **n** est le dernier élément dans le tableau.
4. Si **Non**, la variable **n** reçoit la valeur suivante du tableau et continue au point 2, la ligne juste après le mot clé **For**.  
   n = Array(next)
5. Si **Oui**, le programme continue à la ligne juste après le mot clé **Next**.

Exemple For - Each :

Private Nombres() As Int

Private Somme As Int

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

Somme = 0

For Each n As Int In Nombres

Somme = Somme + n

Next

Même exemple, mais avec une boucle For - Next :

Private Nombres () As Int

Private Somme As Int

Private i As Int

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

Somme = 0

For i = 0 To Nombres.Length - 1

Somme = Somme + Nombres(i)

Next

Cet exemple montre la puissance des boucles For - Each :

For Each lbl As Label In Activity

lbl.TextSize = 20

Next

Même exemple avec une boucle For – Next :

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

### Do - Loop

Plusieurs configurations sont possibles :

Do While test **test**  est n’importe quelle expression conditionnelle

' code Exécute le **code** tant que **test** est égal à **True**

Loop

Do Until test **test**  est n’importe quelle expression conditionnelle

' code Exécute le **code** jusqu’à ce que **test** est égal à **True**

Loop

La boucle **Do While -Loop** fonctionne comme suit :

1. Au début, l’expression **test** est évaluée.
2. Si le résultat est **True**, alors le **code**  est exécuté.
3. Si le résultat est **False,** le programme continue à la ligne juste après le mot clé **Loop**.

La boucle **Do Until -Loop** fonctionne comme suit :

1. Au début, l’expression **test** est évaluée.
2. Si le résultat est **False**, alors le **code**  est exécuté.
3. Si le résultat est **True,** le programme continue à la ligne juste après le mot clé **Loop**.

Il est possible de sortir d’une boucle Do-Loop avec le mot clé **Exit**.

Do While test

' code

If a = 0 Then Exit Si **a = 0**  on sort de la boucle.

' code

Loop

Exemples :

Do Until Loop :

Private i, n As Int

i = 0

Do Until i = 10

' code

i = i + 1

Loop

Do While Loop :

Private i, n As Int

i = 0

Do While i < 10

' code

i = i + 1

Loop

Lire un fichier texte et remplir une liste (List) :

Private lstText As List

Private ligne As String

Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirInternal, "test.txt"))

lstText.Initialize

ligne = tr.ReadLine

Do While line <> Null

lstText.Add(line)

ligne = tr.ReadLine

Loop

tr.Close

**Note :** Différence entre :

B4x VB

Exit Exit Loop

Dans VB le type de boucle, **Loop** dans l’exemple, est ajouté après le mot clé **Exit**.

VB accepte aussi les boucles suivantes, qui ne sont pas implémentées dans B4x.

Do Do

' code ' code

Loop While test Loop Until test

## Subs / routines

Une routine (“Sub”) est une portion de code. Elle peut être de n’importe quelle longueur, a un nom distinctif et une portée définie (dans le sens de portée des variables comme évoqué auparavant). Dans le code B4x, une routine est appelée “Sub”, et est équivalente à une procédure, fonction, méthode et sub dans d’autres langages de programmation. Les lignes de code, à l’intérieur d’une Sub, sont exécutées de la première à la dernière.

Il est déconseillé d’avoir des routines avec beaucoup de code, elles deviennent moins lisibles.

### Déclaration

Une Sub est déclarée de la manière suivante :

Sub **CalculInteret**(Capital As Double, Taux As Double) As Double

Return Capital \* Taux / 100

End Sub

On débute avec le mot clé **Sub**, suivi par le nom, suivi par une liste de paramètres, suivi par le type de retour et se termine avec les mots clé **End Sub**.

Les Subs doivent toujours être déclarées au niveau supérieur du module, on ne peut pas déclarer une Sub à l’intérieur d’une autre Sub.

### Appel d’une Sub

Pour exécuter le code dans une Sub, écrivez simplement le nom de la Sub suivi de la liste des paramètres s’il y a lieu.

Par exemple :

Interet = CalculInteret(1234, 5.2)

Interet Valeur retournée par la Sub.

CalculInteret Nom de la Sub.

1235 Capital, valeur transmise à la Sub.

5.25 Taux, valeur transmise à la Sub.

### Appel d’une Sub depuis un autre module

Une routine déclarée dans un module, avec les mots clé **Public Sub**, est accessible depuis n’importe quel autre module, mais son nom doit être précédé par le nom du module dans lequel est a été déclarée.

Exemple : Si la routine CalculInteret a été déclarée dans le module MonModule alors l’appel à la routine doit être :

Interet = MyModule.CalculInteret(1234, 5.2)

Au lieu de :

Interet = CalculInteret(1234, 5.2)

### Noms

Vous pouvez donner n’importe quel nom, à une routine Sub. Il est conseillé de donner des noms significatifs, comme **CalculInteret** dans l’exemple, de cette manière vous connaissez le but de la routine en lisant le code.

Il n’y a pas de limite au nombre de routines dans le programme, mais vous ne pouvez pas définir deux routines avec le même nom dans un même module.

### Paramètres

On peut transmettre des paramètres à une routine. La liste suit directement le nom de la routine, entre parenthèses.

Les types des paramètres doivent être inclus directement dans la liste.

Sub **CalculInteret**(Capital As Double, Taux As Double) As Double

Return Capital \* Taux / 100

End Sub

Dans B4x, sont transmis par valeur et non par référence.

### Valeur renvoyée

Une routine Sub peut renvoyer une valeur, qui peut être n’importe quel objet.

Le renvoi d’une valeur est effectué avec le mot clé Return.

Le type de la valeur de retour est ajouté après la liste des paramètres.

Sub **CalculInteret**(Capital As Double, Taux As Double) As Double

Return Capital \* Taux / 100

End Sub

## Routines Sub ‘Resumable’

Les routines ‘resumable’ ont été introduites dans B4A v7.00 / B4i v4.00 / B4J v5.50. Elles simplifient drastiquement les opérations asynchrones.  
(Cette fonctionnalité est une variante de ‘stackless [coroutines](https://fr.wikipedia.org/wiki/Coroutine)’.)

Vous trouvez plus d’exemples dans le [forum](https://www.b4x.com/android/forum/threads/b4x-resumable-subs-sleep-wait-for.78601/#content).

La fonctionnalité particulière des ‘resumable subs’ est que l’exécution peut être interrompue, sans interrompre l’exécution du reste du programme, puis reprise plus tard.  
Le programme n’attend pas la reprise du code de la routine, d’autres événements sont générés comme d’habitude.   
  
Toute routine contenant au moins un appel à **Sleep** ou **Wait For** est une ‘resumable sub’.

Dans l’éditeur elles se distinguent par cet indicateur  après la déclaration :

Private Sub **CountDown**(Start As Int)

For i = Start To 0 Step -1

Label1.Text = i

Sleep(1000)

Next

End Sub

### Sleep

Sleep interrompt l’exécution de la routine et la reprend après le temps spécifié.

**Sleep** (Milliseconds As Int) Milliseconds, temps d’interruption en millisecondes.

Exemple :

Sleep(1000)

L’utilisation de Sleep est simple :

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

L’exécution sera interrompue durant 1000 millisecondes puis reprise.  
  
Vous pouvez spécifier Sleep(0) pour une interruption la plus courte. Ceci peut être utilise lorsque vous voulez que l’IHM soit réactualisé. C’est une bonne alternative à DoEvents (qui n’existe pas dans B4J et B4i et doit être évité dans B4A).

Sub **VeryBusySub**

For i = 1 To 10000000

'do something

If i Mod 1000 = 0 Then Sleep(0) 'allow the UI to refresh every 1000 iterations.

Next

Log("finished!")

End Sub

### Wait For

Les langages B4X sont gérés par des événements ([programmation événementielle](https://fr.wikipedia.org/wiki/Programmation_%C3%A9v%C3%A9nementielle)). Des opérations asynchrones fonctionnent en arrière-plan et génèrent des événements lorsque l’opération est terminée.  
Avec le nouveau mot clé **Wait For** vous pouvez gérer les événements à l’intérieur de la routine courante.  
  
Par exemple, le code ci-dessous attend l’événement Ready de GoogleMap (exemple B4J) :

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

MainForm = Form1

MainForm.RootPane.LoadLayout("1") 'Load the layout file.

gmap.Initialize("gmap")

Pane1.AddNode(gmap.AsPane, 0, 0, Pane1.Width, Pane1.Height)

MainForm.Show

Wait For gmap\_Ready '<----------------

gmap.AddMarker(10, 10, "Marker")

End Sub

Un exemple un peu plus compliqué avec FTP :  
Listage de tous les fichiers se trouvant dans un dossier à distance plus leur téléchargement :

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

Lorsque le mot clé **Wait For** est exécuté, la routine est interrompue et le gestionnaire interne des événements prend en charge la reprise de l’exécution du reste du code lorsque l’événement est généré. Si l’événement n’est jamais généré, le reste du code ne sera jamais exécuté. Cela n’entrave pas le fonctionnement du reste du programme qui reste parfaitement réactif.  
Si **Wait For** est appelé à nouveau plus tard, avec le même événement, une nouvelle instance de la routine remplacera la précédente.

Nous allons créer une routine qui télécharge une image et l’attribue à un objet ImageView :

'Mauvais exemple, Ne pas utiliser.

Sub **DownloadImage**(Link As String, iv As ImageView)

Dim job As HttpJob

job.Initialize("", Me) 'notez que le nom parameter n’est plus nécessair.

job.Download(Link)

Wait For JobDone(job As HttpJob)

If job.Success Then

iv.SetImage (job.GetBitmap) 'utiliser iv.Bitmap = job.GetBitmap dans B4A / B4i

End If

job.Release

End Sub

Ceci fonctionne à condition de n’appeler la routine qu’une seule fois (ou plus correctement, si nous appelons la routine à nouveau qu’après que l’opération précédente ne soit terminée).  
Si nous l’appelions comme ci-dessous :

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

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

Seule la deuxième image sera affichée car le deuxième appel à Wait For JobDone va écraser le précédent.  
Ceci nous amène à la deuxième variante de Wait For.  
Pour résoudre ce problème, Wait For peut distinguer les événements basés sur l’émetteur d’événements ‘event sender’.  
Pour cela, on utilise un paramètre optionnel :  
  
*Wait For* (<sender>) <event signature>  
  
Exemple :

'Bon example. A utiliser.

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

Avec le code ci-dessus, chaque instance de la ‘resumable sub’ attend des événements différents et ne sera pas affecté par d’autres appels à la routine.

### Flux du code

Sub **S1**

Log("S1: A")

S2

Log("S1: B")

End Sub

Sub **S2**

Log("S2: A")

Sleep(0)

Log("S2: B")

End Sub

La réponse est:  
S1: A  
S2: A  
S1: B  
S2: B

Chaque fois que Sleep ou Wait For est appelé, la routine courante est interrompue. Ce qui équivaut à appeler Return.

### Attendre qu’une routine ‘resumable sub’ se termine

Si une routine appelle une autre routine ‘resumable’, le code dans la première routine sera exécuté après le premier appel de Sleep ou Wait (dans la deuxième routine).  
  
Si, dans la première routine, vous voulez attendre que l’exécution de la deuxième routine soit terminée, vous pouvez générer un événement dans la deuxième routine et l’attendre dans la première :

Sub **PreimiereSub** 

Log("Première Sub démarrée")

SecondSub

Wait For DeuximenSub\_Termine

Log("Première Sub terminée")

End Sub

Sub **DeuxiemeSub** 

Log("Deuxième Sub démarrée ")

Sleep(1000)

Log("Deuxième Sub terminée")

CallSubDelayed(Me, "DeuxiemenSub\_Termine")

End Sub

Logs :  
Première Sub démarrée  
Deuxième Sub démarrée

Deuxième Sub terminée  
Première Sub terminée  
  
Notes :  
- Il est plus sûr d’utiliser CallSubDelayed au lieu de CallSub. CallSub ne fonctionnera pas si la deuxième routine n’est jamais interrompue (par exemple si Sleep est seulement appelé par une condition).  
- Le code ci-dessus admet que PremiereSub n’est jamais appelé à nouveau tant que son exécution ne soit terminée.

### DoEvents

A partir de B4A version 7.00, l’avertissement suivant sera affiché pour tout appel à DoEvents :  
*DoEvents est déprécié. It can lead to stability issues. Use Sleep(0) instead (if really needed).*  
  
La fonction de DoEvents consistait à permettre des mises à jour de l’interface utilisateur lorsque le processus principal était occupé. DoEvents, qui partage la même implémentation que les dialogues modaux, est une implémentation à bas niveau. Il accède à la queue des messages du processus et exécute quelques-uns des messages en attente.  
  
Avec l’évolution d’Android, la gestion de la queue des messages est devenue plus sophistiquée et plus fragile.  
Les raisons de la dépréciation de DoEvents sont :  
  
1. C’est une source majeure d’instabilités. Cela peut conduire à des crashs difficiles à déboguer ou à des messages ANR (application not responding, l’application ne répond pas). Notez que ceci est aussi valable pour les dialogues modaux (tel que Msgbox ou InputList).  
2. Il existe de meilleures façons de maintenir le processus principal libre. Par exemple, en utilisant pour les bases de données SQL, les méthodes asynchrones [asynchronous SQL methods](https://www.b4x.com/android/forum/threads/79532/#content) au lieu des méthodes synchrones.  
3. DoEvents ne fait pas ce que beaucoup de développeur attendant que ça fasse. Car DoEvents ne traite que des messages d’interface utilisateur, la plupart des événements ne peuvent pas être générés à partir d’un appel à DoEvents.  
4. Maintenant on peut appeler Sleep pour interrompre l’exécution de la routine courante et la reprendre après que les messages d’attente soient traités. [Sleep implementation](https://www.b4x.com/android/forum/threads/78601/#content) est totalement différent de DoEvents. Il ne bloque pas le processus. Il le libère tout en préservant l’état de la routine.  
Contrairement à DoEvents qui ne traite que les messages de l’interface utilisateur, avec Sleep tous les messages seront générés.  
(Notez qu’utiliser Wait For et attendre un événement est préférable à utiliser Sleep dans une boucle).  
  
Malgré tout, DoEvents est maintenu et les applications existantes fonctionnent exactement comme auparavant.

### Dialogues / Dialogs

Dialogues modaux = dialogues qui bloquent le processus principal jusqu’à ce que leur traitement ne soit terminé.  
  
Comme déjà évoqué dans le chapitre précédent, les dialogues modaux utilisent la même implémentation que DoEvents. Il est donc recommandé de passer à l’utilisation des nouveaux dialogues asynchrones. Le passage à [Wait For](https://www.b4x.com/android/forum/threads/78601/#content) est vraiment un changement simple :

Au lieu de :

Dim rep As Int = Msgbox2("Effacer?", "Titre", "Oui", "Abandon", "Non", Null)

If rep = DialogResponse.POSITIVE Then

'...

End If

Vous devez utiliser :

Msgbox2Async("Effacer?", "Titre", "Oui", "Abandon", "Non", Null, False)

Wait For Msgbox\_Result (Result As Int)

If Result = DialogResponse.POSITIVE Then

'...

End If

*Wait For* ne bloque pas le processus principal. Il mémorise l’état actuel de la routine et la libère. L'exécution du code sera reprise lorsque l’utilisateur clique sur un des boutons du dialogue.  
Les autres méthodes asynchrones similaires sont : MsgboxAsync, InputListAsync et InputMapAsync.  
  
A l’exception de MsgboxAsync, les nouvelles méthodes ont un nouveau paramètre *Cancelable*. S’il est égal à True, la boite de dialogue peut être abandonnée en cliquant sur le bouton Retour ‘Back’ ou en dehors de la boite de dialogue. Ce qui est le comportement par défaut des méthodes anciennes.  
  
Comme l’exécution du code continue pendant que la boite de dialogue asynchrone est affichée, il est possible que d’autres boites de dialogue soient affichées en même temps.  
Si c’est le cas, vous devez définir un paramètre de filtre d’émetteur, *sf* dans l’exemple, dans l’appel de Wait For :

Dim sf As Object = Msgbox2Async("Effacer?", "Titre", "Oui", "Abandon", "Non", Null, False)

Wait For (sf) Msgbox\_Result (Result As Int)

If Result = DialogResponse.POSITIVE Then

'...

End If

Ceci autorise l’affichage de messages multiples avec un traitement correct des événements.

### SQL avec Wait For

La nouvelle fonctionnalité des ‘resumable subs’, rend le travail plus simple avec de grandes quantités de données avec une influence minime sur la réactivité du programme.  
  
La nouvelle méthode pour insérer des données est :

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)

Les étapes sont :  
- Appelez AddNonQueryToBatch pour chaque commande qui doit être effectuée.   
- Exécutez la commande avec ExecNonQueryBatch. C’est une méthode asynchrone. Les commandes seront exécutées en arrière-plan et l’événement NonQueryComplete sera généré lorsque l’exécution est terminée.  
- Cet appel renvoie un objet qui peut être utilisé comme paramètre de filtre d’émetteur. Ceci est important car il pourrait y avoir plusieurs lots d’exécutions en arrière-plan. Avec le paramètre de filtre d’émetteur l’événement sera traité par l’appel Wait For correctement dans tous les cas.  
- Notez que SQL1.ExecNonQueryBatch commence et termine une transaction en interne.

#### Requêtes / Queries

Dans la plupart des cas les requêtes sont rapides et devraient donc être appelées de manière synchrone SQL1.ExecQuery2. Néanmoins, s’il y a une requête lente vous pouvez passer à 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

Comme dans le cas précédent, la méthode ExecQueryAsync renvoie un objet qui peut être utilise comme paramètre de filtre d’émetteur.  
  
Conseils :  
1. L’objet ResultSet dans B4A est une extension de l’objet Cursor. Vous pouvez le changer en Cursor si vous préférez. L’avantage dans l’utilisation de ResultSet est sa compatibilité avec B4J et B4i.  
2. Si le nombre de lignes renvoyé per la requête est grand, la boucle Do While peut devenir lents en mode ‘Debug’. Vous pouvez la rendre plus rapide en la déplaçant dans une routine différente et en ‘nettoyant’ le projet (Ctrl + P) :

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

Ceci est lié à une optimisation du débogueur qui est actuellement désactivée pour les routines ‘resumable subs’.  
La performance des deux solutions est la même en mode ‘Release’.

#### B4J

- Nécessite jSQL v1.50+ (<https://www.b4x.com/android/forum/threads/updates-to-internal-libaries.48274/#post-503552>).  
- Il est recommandé de définir le mode journal à: [https://www.b4x.com/android/forum/t...ent-access-to-sqlite-databases.39904/#content](https://www.b4x.com/android/forum/threads/webapp-concurrent-access-to-sqlite-databases.39904/#content)

### Notes & Conseils

* Les routines ‘resumable subs’ ne peuvent pas renvoyer des valeurs.
* L’influence des ‘resumable subs’ sur la performance du programme devrait être insignifiante en mode ‘Release’ dans la majorité des cas. L’influence pourrait être plus importante en mode ‘Debug’. (Si ça devenait un problème déplacez les parties lentes du code dans d’autres routines qui seront appelées depuis la ‘resumable sub’.)
* Les gestionnaires d’événements Wait For sont prioritaires sur les gestionnaires d’événements réguliers.
* Les ‘resumable subs’ ne créent pas de processus additionnels. Le code est exécuté dans le processus principal, ou processus de gestion pour les solutions serveur.

## Événements (Events)

En programmation orienté objet nous avons des objets qui peuvent réagir à différentes actions de l’utilisateur appelées événements (events).

Le nombre et les types d’événements qu’un objet peut générer dépend du type de l’objet.

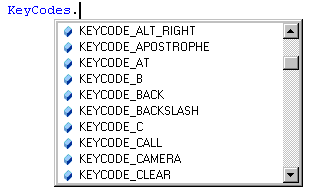
### B4A

Les objets d’interface utilisateur sont appelés 'view' (vue) dans Android.

Résumé des événements pour les différents objets ‘view’ :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Événements / Events** | | | | | | | | | | | | | | | | | |
| **Views** | Click | LongClick | Touch | Down | Up | KeyPress | KeyUp | ItemClick | ItemLongClick | CheckedChange | EnterPressed | FocusChanged | TextChanged | ScrollChanged | ValueChanged | TabChanged | OverrideUrl | PageFinished |
| Activity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Button |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CheckBox |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| EditText |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| HorizontalScrollView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ImageView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Label |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ListView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Panel |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RadioButton |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ScrollView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SeekBar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Spinner |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TabHost |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ToggleButton |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WebView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Les événements les plus courants sont :

* **Click** Événement généré lorsque l’utilisateur presse, clique sur la ‘view’.  
  Exemple :  
  Sub **Button1\_Click**  
   ' votre code  
  End Sub
* **LongClick**   
  Événement généré lorsque l’utilisateur clique sur la ‘view’ et maintient pour un moment.  
  Exemple :  
  Sub **Button1\_LongClick**  
   ' votre code  
  End Sub
* **Touch** (Action As Int, X As Float, Y As Float)  
  Evénement généré lorsque l’utilisateur touche la ‘view’.   
    
  Trois différentes actions sont traitées:  
  - Activity.Action\_DOWN, l’utilisateur touche l’écran.  
  - Activity.Action\_MOVE, l’utilisateur déplace son doigt sur l’écran sans le lever.  
  - Activity.Action\_UP, l’utilisateur quitte l’écran.  
    
  Les coordonnées X et Y de la position du doigt sont donnés en pixels.  
    
  Exemple :  
  Sub **Activity\_Touch** (Action As Int, X As Float, Y As Float)  
   Select Action  
   Case Activity.ACTION\_DOWN  
   ' votre code pour Action = 0 donc DOWN   
   Case Activity.ACTION\_MOVE  
   ' votre code pour Action = 2 donc MOVE   
   Case Activity.ACTION\_UP  
   ' votre code pour Action = 1 donc UP   
   End Select  
  End Sub
* **CheckChanged** (Checked As Boolean)  
  Evénement généré lorsque l’utilisateur clique sur une CheckBox ou un RadioButton.  
  Checked (coché) est égal à True si la ‘view’ est cochée et False si non coché.  
    
  Exemple :  
  Sub **CheckBox1\_CheckedChange**(Checked As Boolean)  
   If Checked = True Then  
   ' votre code pour coché Checked = True  
   Else  
   ' votre code pour non coché Checked = False  
   End If  
  End Sub
* **KeyPress** (KeyCode As Int) As Boolean  
  Evénement généré lorsque l’utilisateur presse sur une touche physique ou virtuelle.  
  KeyCode (code de la touche) est le code de la touche pressée, on peut obtenir ces codes avec le mot clé KeyCodes.  
    
    
    
  L’événement peut renvoyer soit :  
  - True, l’événement est 'consommé', considéré par le système d’exploitation comme déjà exécuté et aucune autre action n’est prise.  
  - False, l’événement n’est pas 'consommé' et est transmis au système pour d’autres actions.   
    
  Exemple:  
    
  Sub **Activity\_KeyPress**(KeyCode As Int) As Boolean  
   Private Rep As Int  
   Private Txt As String  
     
   If KeyCode = KeyCodes.KEYCODE\_BACK Then ' Teste si KeyCode est BackKey  
   Txt = "Voulez-vous vraiment quitter programme ?"  
   Rep = Msgbox2(Txt,"A T T E N T I O N","Oui","","Non",Null)' MessageBox  
   If Rep = DialogResponse.POSITIVE Then ' Si Rep = Oui   
   Return False ' Return = False l’événement ne sera pas consommé  
   Else ' nous quittons le programme  
   Return True ' Return = True l’événement sera consommé pour  
   End If ' éviter de quitter le programme  
   End If  
  End Sub

### B4i

Les objets d’interface utilisateur s’appellent 'view’ dans iOS.

Résumé des événements des différents objets ‘view’ :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Événements / Events** | | | | | | | | | | | | | | |
| **Views** | Click | LongClick | BeginEdit | EndEdit | EnterPressed | TextChanged | Touch | Resize | ScrollChanged | ValueChanged | ItemSelected | IndexChanged | OverrideUrl | PageFinished |
| Button |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TextField |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TextView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ImageView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Label |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Panel |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ScrollView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Slider |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Picker |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Stepper |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Switch |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SegmentedControl |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Slider |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Stepper |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WebView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Les événements les plus courants sont :

* **Click** Événement généré lorsque l’utilisateur presse, clique sur une ‘view’.  
  Exemple :  
  Private Sub **Button1\_Click**  
   ' votre code  
  End Sub
* **LongClick**   
  Événement généré lorsque l’utilisateur clique sur la ‘view’ et maintient pour un moment.  
  Exemple :  
  Private Sub **Button1\_LongClick**  
   ' votre code  
  End Sub
* **Touch** (Action As Int, X As Float, Y As Float)  
  Événement généré lorsque l’utilisateur touche un Panel sur l’écran.   
    
  Trois actions différentes sont traitées :  
  - Panel.ACTION\_DOWN, l’utilisateur touche l’écran.  
  - Panel.ACTION\_MOVE, l’utilisateur déplace son doigt sans le lever de l’écran.  
  - Panel.ACTION\_UP, l’utilisateur lâche l’écran.  
    
  Les coordonnées X et Y de la position du doigt sont donnés en Points et pas en Pixels.  
    
  Exemple :  
  Private Sub **Panel\_Touch** (Action As Int, X As Float, Y As Float)  
   Select Action  
   Case Panel.ACTION\_DOWN  
   ' Votre code pour l’action DOWN  
   Case Panel.ACTION\_MOVE  
   ' Votre code pour l’action MOVE  
   Case Panel.ACTION\_UP  
   ' Votre code pour l’action UP  
   End Select  
  End Sub

### B4J

Les objets d’interface utilisateur s’appellent 'node' (nœud) dans Java.

Résumé des événements des différents objets ‘node’ :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Événements / Events** | | | | | | | | | | | | | | | | |
| **Nodes** | Action | FocusChanged | MouseClicked | MouseDragged | MouseMoved | MousePressed | MouseReleased | Resize | CheckedChange | SelectedIndexChangedd | ValueChange | SelectedChange | H / VScrollChanged | TabChanged | TextChanged | PageFinished |
| Button |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Canvas |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CheckBox |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ChoiceBox |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ComboBox |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ImageView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Label |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ListView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pane |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| RadioButton |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ScrollPane |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Slider |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Spinner |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TabPane |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TextArea |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TextField |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ToggleButton |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| WebView |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Les événements les plus courants sont :

* **Action**   
  Événement généré lorsque l’utilisateur clique sur un ‘node’ (Button ou TextField).  
  Exemple :  
  Private Sub **Button1\_Action**  
   ' votre code  
  End Sub
* **FocusChanged** (Has Focus As Boolean)   
  Événement généré lorsque le ‘node’ reçoit ou perd le Focus.  
  Exemple :  
  Private Sub **TextField1\_FocusChanged** (HasFocus As Boolean)  
   ' votre code  
  End Sub
* **MouseClicked** (EventData As MouseEvent)  
  Événement généré lorsque l’utilisateur clique sur un ‘node’.   
  Exemple :  
  Private Sub **Pane1\_MouseClicked** (EventData As MouseEvent)  
   ' votre code  
  End Sub
* **MouseDragged** (EventData As MouseEvent)  
  Événement généré lorsque l’utilisateur se déplace sur un ‘node’ avec un bouton de la souris pressé.   
  Similaire à ACTION\_MOVE dans les événement Touch dans B4A et B4i.
* Exemple :  
  Private Sub **Pane1\_MouseDragged** (EventData As MouseEvent)  
   ' votre code  
  End Sub
* **MouseMoved** (Eventât As MouseEvent)  
  Événement généré lorsque l’utilisateur se déplace sur un ‘node’ sans bouton de la souris pressé.   
  Exemple :  
  Private Sub **Pane1\_MouseMoved** (EventData As MouseEvent)  
   ' votre code  
  End Sub
* **MousePressed** (EventData As MouseEvent)  
  Événement généré lorsque l’utilisateur presses sur un ‘node’.   
  Similaire à ACTION\_DOWN dans les événements Touch dans B4A et B4i.  
  Exemple :  
  Private Sub **Pane1\_MousePressed** (EventData As MouseEvent)  
   ' votre code  
  End Sub
* **MouseReleased** (EventData As MouseEvent)  
  Événement généré lorsque l’utilisateur relâche le ‘node’.  
  Similaire à ACTION\_UP dans les événements Touch dans B4A et B4i.   
  Exemple :  
  Private Sub **Pane1\_MouseReleased** (EventData As MouseEvent)  
   ' votre code  
  End Sub
* **MouseEvent** objet renvoyé par les événements Mouse (souris).  
  Données dans l’objet MouseEvent :
* **ClickCount** Nombre de clics associés avec cet événement.
* **Consume** Consomme l’événement actuel et évite qu’il soit traité par le  
   ‘node’ parent.
* **MiddleButtonDown** Égal à True si le bouton du milieu est pressé lors de  
   l’événement.
* **MiddleButtonPressed** Égal à True si le bouton du milieu a généré l’événement.
* **PrimaryButtonDown** Égal à True si le bouton primaire est pressé lors de  
   l’événement.
* **PrimaryButtonPressed** Égal à True si le bouton primaire a généré de l’événement.
* **SecondaryButtonDown** Égal à True si le bouton secondaire est pressé lors de  
   l’événement.
* **SecondaryButtonPressed** Égal à True si le bouton secondaire a généré de l’événement.
* **X** Coordonnée X par rapport aux bords du ‘node’.
* **Y** Coordonnée Y par rapport aux bords du ‘node’.

### B4R

Dans B4R, les objets Pin et [Timer](#_Timers) sont les seul à générer des événements :

* Pin  
  **StateChanged** (State As Boolean) Événement généré lorsque une ‘pin’ change d’état.  
    
  Exemple :  
  Sub **Pin1\_StateChanged(State As Boolean)**  
   ' votre code  
  End Sub
* Timer  
  **Tick** Événement généré pour chaque intervalle de temps donnée.  
    
  Exemple :  
  Sub **Timer1\_Tick**  
   ' votre code  
  End Sub

### Résumé des objets d’interfaces utilisateur

Le tableau ci-dessous montre un résumé des objets d’interface utilisateur ‘standards’.

Il montre les différences entre les trois systèmes d’exploitation.

Certains objets qui n’existent pas en tant qu’objet standard peuvent exister en tant qu’objets personnalisés (CustomView) dans d’autres systèmes d’exploitation.

Vous pouvez chercher dans le forum.

|  |  |  |  |
| --- | --- | --- | --- |
| View / node | B4A | B4i | B4J |
| Activity |  |  |  |
| Button |  |  |  |
| CheckBox |  |  |  |
| EditText |  |  |  |
| HorizontalScrollView |  |  |  |
| ImageView |  |  |  |
| Label |  |  |  |
| ListView |  |  |  |
| Panel |  |  |  |
| RadioButton |  |  |  |
| ScrollView |  |  |  |
| SeekBar |  |  |  |
| Spinner |  |  |  |
| TabHost |  |  |  |
| ToggleButton |  |  |  |
| WebView |  |  |  |
| TextField |  |  |  |
| TextView |  |  |  |
| ScrollView different de B4A 2D |  |  |  |
| Slider |  |  |  |
| Picker |  |  |  |
| Stepper |  |  |  |
| Switch |  |  |  |
| SegmentedControl |  |  |  |
| Canvas un node à part entière |  |  |  |
| ChoiceBox |  |  |  |
| ComboBox |  |  |  |
| Pane similaire à Panel dans B4A et B4i |  |  |  |
| ScrollPane similaire à ScrollView |  |  |  |
| TabPane |  |  |  |
| TextArea |  |  |  |

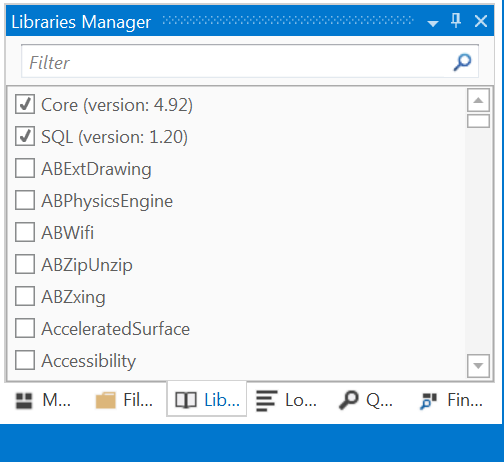
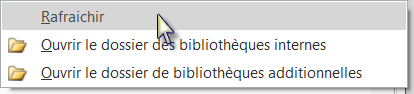
## Bibliothèques (Libraries)

Les bibliothèques ajoutent plus d’objets et fonctionnalités à B4x.

Certaines sont livrées avec les produits B4x et font partie intégrante du système de développement standard, appelées ‘bibliothèques standard’.

D’autres, appelées ‘bibliothèques additionnelles’, souvent développées par des utilisateurs, peuvent être téléchargées du forum (seulement par des utilisateurs enregistrés) pour ajouter des fonctionnalités supplémentaires aux environnement de développement B4x.

Si vous avez besoin d’une bibliothèque, vous devez :

* Vérifier dans l’onglet Bibliothèques si vous avez déjà cette bibliothèque.
* Pour les bibliothèques additionnelles, vérifier que c’est la version la plus récente.  
  Vous pouvez vérifier les versions dans les pages ‘Documentation’ dans le forum.  
  [B4A](https://www.b4x.com/android/documentation.html), [B4i](https://www.b4x.com/b4i/documentation.html), [B4J](https://www.b4x.com/b4j/documentation.html), [B4R](https://www.b4x.com/b4r/documentation.html)  
  Pour trouver les fichiers d’une bibliothèque utilisez une requête comme <http://www.b4x.com/search?query=betterdialogs+library>   
  dans l’explorateur internet.
* Si la bibliothèque existe vous devez la cocher dans l’onglet Gestionnaire des Bibliothèques.  
    
  
* Si elle n’existe pas, téléchargez-la, dézippez-la et copiez les fichiers < NomBibilithèque>.jar et < NomBibilithèque>.xml dans le dossier des bibliothèques additionnelles.
* Cliquez avec le bouton droit de la souris dans la liste des bibliothèques puis cliquez sur  et cochez la bibliothèque dans la liste pour la sélectionner.  
    
  

### Bibliothèques standard

Les bibliothèques B4x standard sont sauvées dans les dossiers Libraries dans les dossiers des programmes.

Généralement dans :

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

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

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

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

### Dossier Bibliothèques additionnelles

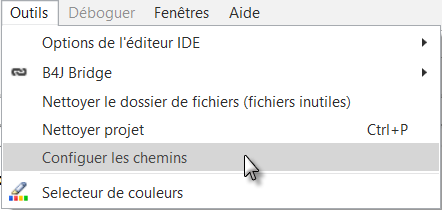
Pour les fichiers des bibliothèques additionnelles il est judicieux de créer un dossier spécifique.

Par exemple :

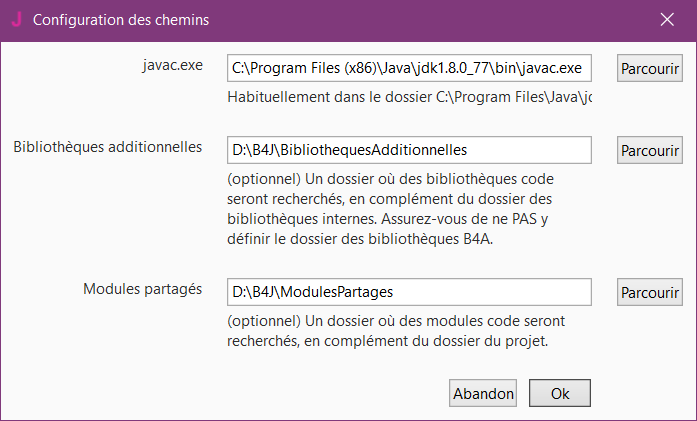
C:\B4A\BiblothequesAddidionnelles, C:\B4i\ BiblothequesAddidionnelles, C:\B4J\ BiblothequesAddidionnelles, C:\B4R\ BiblothequesAddidionnelles

Lorsque vous installez une nouvelle version d’un des produits B4x, toutes les bibliothèques standard sont automatiquement mises à jour, ce qui n’est pas le cas des bibliothèques additionnelles. L’avantage du dossier spécifique est que vous n’avez pas à vous soucier des bibliothèques additionnelles car le dossier spécifique n’est pas affecté lors de l’installation d’une nouvelle version.

Les bibliothèques additionnelles ne sont pas mises à jour automatiquement lors d’une mise à jour de B4x.

Lorsque l’éditeur démarre, il cherche d’abord dans le dossier des bibliothèques B4x standard puis dans le dossier des bibliothèques additionnelles.

Vous pouvez définir le dossier des bibliothèques additionnelles dans l’éditeur dans le menu Outils / Configurer les chemins.



Exemple pour B4J. C’est similaire pour las autres produits.

Entrez le nom du dossier et cliquez sur .

Ou cliquez sur  pour le chercher.

### Téléchargement et mise à jour d’une bibliothèque

Une liste des bibliothèques officielles et additionnelles avec des liens sur leur documentation peut être trouvée sur les pages ‘Documentation’ sur le site :

Page Documentation B4A : [List of Libraries](https://www.b4x.com/android/documentation.html).

Page Documentation B4i : [List of Libraries](https://www.b4x.com/android/documentation.html).

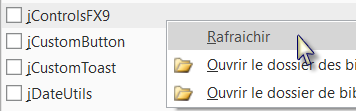
Page Documentation B4J : [List of Libraries](https://www.b4x.com/android/documentation.html).

Page Documentation B4R : [List of Libraries](https://www.b4x.com/android/documentation.html).

Pour trouver les fichiers des bibliothèques, utilisez une requête comme <http://www.b4x.com/search?query=betterdialogs+bibliothèque>

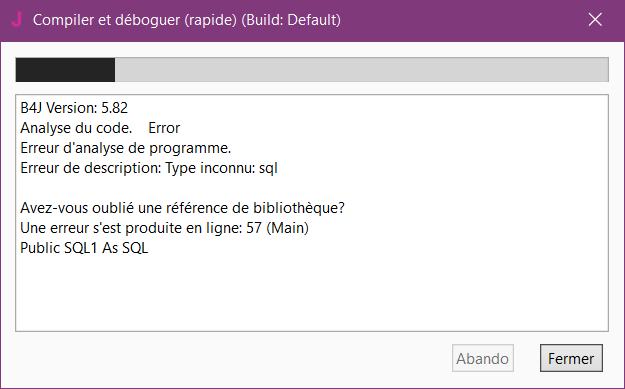
dans l’explorateur internet.

Pour charger ou mettre à jour une bibliothèque suivez les étapes ci-dessous :

* Téléchargez le fichier quelque part sur votre disque.
* Dézippez le.
* Copiez les fichiers xxx.jar et xxx.xml vers
  + Le dossier Library de B4x pour les bibliothèques standard
  + Le dossier [BibliothèquesAddionnelles](#_Additional_libraries_folder) pour une bibliothèque additionnelle.
* Cliquez avec le bouton droit dans le [Gestionnaire des Bibilothèques](#_Libs) de l’éditeur puis cliquez   
    
  sur  et cochez la pour la sélectionner.

### Message d’erreur "Avez-vous oublié une référence de bibliothèque ?"

Si vous obtenez un message similaire à celui ci-dessous, ça veut dire que vous avez oublié de cocher la bibliothèque spécifiée ou que vous ne l’avez pas !



## Manipulations de textes / objet String

### B4A, B4i, B4J

B4A, B4i et B4J autorisent des manipulations de texte (objet String) comme les autres langages Basic avec quelques différences.

Ces manipulations peuvent être opérées directement sur l’objet String.

Exemple :

txt = "123,234,45,23"

txt = txt.Replace(",", ";")

Résultat : 123;234;45;23

Les différentes fonctions sont :

* **CharAt(Index)** Renvoie le caractère à l’index donné.
* **CompareTo(Other)** Compare lexicographiquement deux objets String.
* **Contains(SearchFor)** Teste si le texte contient le texte donné dans SearchFor.
* **EndsWith(Suffix)** Renvoie True si le texte se termine avec le texte Suffix.
* **EqualsIgnoreCase(Other)** Renvoie True si les deux textes sont identiques indépendamment de la casse.
* **GetBytes(Charset)** Encode le texte avec le code caractères donné dans Charset en un nouveau texte.
* **IndexOf(SearchFor)** Renvoie l’index de la première occurrence du texte donné dans SearchFor. Le premier index est 0. Renvoie -1 si le texte n’est pas trouvé.
* **IndexOf2(SearchFor, Index)** Renvoie l’index de la première occurrence du texte donné dans SearchFor. Débute la recherche à partir de l’index donné.   
  Le premier index est 0. Renvoie -1 si le texte n’est pas trouvé.
* **LastIndexOf(SearchFor)** Renvoie l’index de la première occurrence du texte donné dans SearchFor. Débute la recherche à partir de la fin du texte et avance vers le début.   
  Le premier index est 0. Renvoie -1 si le texte n’est pas trouvé.
* **LastIndexOf2(SearchFor)** Renvoie l’index de la première occurrence du texte donné dans SearchFor. Débute la recherche à partir de l’index donné et avance vers le début.   
  Le premier index est 0. Renvoie -1 si le texte n’est pas trouvé.
* **Length** Renvoie la longueur, nombre de caractères, du texte.
* **Replace(Target, Replacement)** Renvoie un nouvel objet String résultant du remplacement de tous les textes donnés dans Target avec le texte donné dans Replacement.
* **StartsWith(Prefix)** Renvoie True si le texte débute avec le texte donné dans Prefix.
* **Substring(BeginIndex)** Renvoie un nouvel objet String qui est un texte tronqué du texte d’origine, débutant avec le caractère à l’index donné dans BeginIndex jusqu’à la fin.
* **Substring2(BeginIndex, EndIndex)** Renvoie un nouvel objet String qui est un texte tronqué du texte d’origine, débutant avec le caractère à l’index donné dans BeginIndex jusqu’au caractère à l’index EndIndex avant la fin.  
  Notez que EndIndex est l’index de fin et non le nombre de caractères comme dans certains autres langages.
* **ToLowerCase** Renvoie le texte en minuscules.
* **ToUpperCase** Renvoie le texte en majuscules.
* **Trim** Renvoie un copie du texte en éliminant tous les caractères vides au début et à la fin.

**Note :** Les fonctions ‘string’ sont sensibles à la casse.

Si vous voulez utiliser ces fonctions insensibles à la casse, vous devez utiliser soit ToLowerCase ou ToUpperCase.

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

### B4R

B4R ne supporte pas les manipulations de texte comme les autres langages Basic.

Ces manipulations peuvent être effectuées avec l’objet ByteConverter de la bibliothèque rRandomAccesFile.

Les objets String dans B4R sont différents des ceux dans les autres langages B4x. Les raisons de ces différences sont :

* Mémoire très limitée.
* Manque d’encodage Unicode.

Un objet String dans B4R est la même chose que char\* string dans le langage C.

C’est un tableau d’octets (bytes) avec un octet 0 à la fin.  
La nécessité du 0 à la fin rend impossible la création de textes partiels sans le copier à une nouvelle adresse.

**Pour cette raison, les tableaux d’octets sont préférables aux objets String**.

Les différentes fonctions de manipulation sur les objets String peuvent être effectuées sur des tableaux d’octets.  
  
Convertir un objet String en un tableau d’octets est très simple et ne nécessite pas de copie en mémoire. Le compilateur le fait automatiquement si nécessaire :

Private b() As Byte = "abc" 'équivalent à Private b() As Byte = "abc".GetBytes

Seules les deux fonctions ci-dessous sont supportées :

* **GetBytes(Charset)** Renvoie un tableau d’octets du contenu de l’objet String.  
   Notez que les deux occupent le même espace mémoire
* **Length** Renvoie la longueur, nombre de caractères, du texte.

**Méthodes pour l’objet String**  
  
Les méthodes pour l’objet String se trouvent dans l’objet ByteConverter dans la bibliothèque rRandomAccessFile.  
  
Elles sont similaires à celles des autres langages B4x :

Private Sub AppStart  
   Serial1.Initialize(115200)  
   Log("AppStart")     
   Dim bc As ByteConverter  
   Log("IndexOf: ", bc.IndexOf("0123456", "3")) 'IndexOf: 3  
   Dim b() As Byte = " abc,def,ghijkl "  
   Log("Substring: ", bc.SubString(b, 3)) 'Substring: c,def,ghijkl  
   Log("Trim: ", bc.Trim(b)) 'Trim: abc,def,ghijkl  
   For Each s() As Byte In bc.Split(b, ",")  
     Log("Split: ", s)  
     'Split: abc  
     'Split: def  
     'Split: ghijkl  
   Next  
   Dim c As String = JoinStrings(Array As String("Number of millis: ", Millis, CRLF, "Number of micros: ", Micros))  
   Log("c = ", c)  
   Dim b() As Byte = bc.SubString2(c, 0, 5)  
   b(0) = Asc("X")  
   Log("b = ", b)  
   Log("c = ", c) 'le premier caractère sera X  
End Sub

Notez comment des objets String et des tableaux d’octets peuvent être utilisés vu que le compilateur les convertit automatiquement.  
  
A l’exception de JoinStrings, aucune des méthodes ci-dessus ne fait une copie du texte original.  
Ça veut dire que si on modifie le tableau renvoyé, comme trois dernières lignes, on modifie aussi le tableau d’origine.

Cela se produira également avec des objets String qui partagent tous le même bloc mémoire :

Private Sub AppStart

Serial1.Initialize(115200)

Log("AppStart")

Dim bc As ByteConverter

Dim b() As Byte = bc.Trim("abcdef ")

b(0) = Asc("M") 'cette ligne change aussi le contenu de l’objet String

Dim s as String = "abcdef "

Log(s) 'Mbcdef  
End Sub

Manipulations de texte dans l’objet ByteConverte :

* **EndsWith(Source As Byte(), Suffix As Byte())**   
  Renvoie True si le texte se termine par le texte donné dans Suffix.
* **IndexOf(Source As Byte(), SearchFor As Byte())**   
  Renvoie l’index de la première occurrence du texte donné dans SearchFor dans le texte donné dans Source.
* **IndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**   
  Renvoie l’index de la première occurrence du texte donné dans SearchFor dans le texte donné dans Source. Débute la recherche à partir de l’Index donné.
* **LastIndexOf(Source As Byte(), SearchFor As Byte())**   
  Renvoie l’index de la première occurrence du texte donné dans SearchFor dans le texte donné dans Source. Débute la recherche depuis la fin du texte.
* **LastIndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**   
  Renvoie l’index de la première occurrence du texte donné dans SearchFor dans le texte donné dans Source. Débute la recherche depuis l’Index donné et avance vers le début.
* **StartsWith(Source As Byte(), Prefix As Byte())**   
  Renvoie True si le texte débute avec le texte donné dans Prefix.
* **Substring(Source As Byte(), BeginIndex As UInt)**   
  Renvoie un nouvel objet String qui est un texte tronqué du texte d’origine, débutant avec le caractère à l’index donné dans BeginIndex jusqu’à la fin.
* **Substring2(Source As Byte(), BeginIndex As UInt, EndIndex As UInt)**    
  Renvoie un nouvel objet String qui est un texte tronqué du texte d’origine, débutant avec le caractère à l’index donné dans BeginIndex jusqu’au caractère à l’index EndIndex avant la fin.
* **Trim(Source As Byte())**   
  Renvoie un copie du texte en éliminant tous caractères vides au début et à la fin.

## Formatage de nombres

### B4A, B4i, B4J

Formatage de nombres, afficher des nombres sous forme de texte avec différents formats.

Il existe deux mots clé :

* **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) des variables peuvent être utilisées.  
  NumberFormat(Value + 10, 3 ,0) des opérations arithmétiques peuvent être utilisées.  
  NumberFormat((lblscore.Text + 10), 0, 0) si un des nombres est un objet String mettez-le entre parenthèses.
* **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

### B4R

**Formatage de nombres**, affiche des nombres sous forme de texte dans différents formats :

* **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) des variables peuvent être utilisées.  
  NumberFormat(Value + 10, 3 ,0) des opérations arithmétiques peuvent être utilisées.  
  NumberFormat((lblscore.Text + 10), 0, 0) si un des nombres est un objet String mettez-le entre parenthèses.

## Timers / Minuteurs

Les objets Timer (minuteur) génèrent des événements Tick à des intervalles spécifiés. L’utilisation de Timers est une bonne alternative à de longues boucles, car ils permettent au système d’exploitation de traiter d’autres événements ou messages.  
Notez que les événements ne seront pas générés lorsque l’interface utilisateur est occupée à exécuter d’autres codes.  
Dans B4A, les événements ne sont pas générés lorsque l’Activity est désactivée, ou si un dialogue (comme Msgbox) est visible.  
Il est aussi important, dans B4A, de désactiver les Timers quand l’Activity est mise en veille et de les réactiver lorsque l’Activity reprend dans Activity\_Resume.

Un Timer a :

* Trois paramètres.
  + **Initialize** Initialise le Timer avec deux paramètres, EventName (nom générique de l’événement) et l’intervalle en millisecondes.   
    Timer1.Initialize(EventName As String, Interval As Long)  
    Ex : Timer1.Initialize("Timer1", 1000)
  + **Interval** Définit l’intervalle du Timer en millisecondes.  
    Timer1. Interval = Intervalle  
    Ex : Timer1.Interval = 1000, correspond à 1 seconde
  + **Enabled** Active ou désactive le Timer. **Vaut False par défaut.**  
    Ex : Timer1.Enabled = True
* Un événement
  + **Tick** La routine Tick est appelée tous les intervalles de temps.  
    Ex : Sub Timer1\_Tick

**Les Timer doivent être déclarés dans la routine Process\_Globals.**

Sub Process\_Globals

Public Timer1 As Timer

**Mais ils doivent être initialisée dans une des routines ci-dessous dans le module dans lequel la routine Tick est utilisée.**

**B4A :** routine Activity\_Create

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

If FirstTime = True Then

Timer1.Initialize("Timer1", 1000)

End If

**B4i :** routine Application\_Start

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

Timer1.Initialize("Timer1", 1000)

**B4J :** routine AppStart

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

Timer1.Initialize("Timer1", 1000)

**B4R :** routineAppStart

Private Sub **AppStart**

Timer1.Initialize("Timer1", 1000)

Et la routine événement Timer\_Tick.

Cette routine sera appelée par le système d’exploitation toutes les secondes (1000 millisecondes).

Private Sub **Timer1\_Tick**

' Faire quelque chose

End Sub

## Fichiers B4A, B4i, B4J

Beaucoup d’applications ont besoin d’accéder à des données stockées de manière permanente. Les deux types de stockage les plus courants sont les fichiers et les bases de données.

Android et iOS ont leur propre système de fichiers. Les programmes B4A ni les programmes B4i ont accès au système de fichiers de Windows.

Pour ajouter des fichiers à votre projet, vous devez les ajouter dans l’éditeur dans l’onglet Gestionnaire de fichiers. Ces fichiers seront ajoutés dans le dossier Files de votre projet.

### Objet File (fichiers)

L’objet **File** contient un certain nombre de fonctions pour travailler avec des fichiers.

**Emplacements de fichiers**

- Il y a quelques emplacements importants où vous pouvez lire ou écrire des fichiers.

**File.DirAssets**  
- Le dossier *DirAssets* contient les fichiers qui ont été ajoutés dans l’onglet Gestionnaire de fichiers dans l’éditeur, c’est le dossier *Files* du projet.

**Ces fichiers sont en lecture seule !**

Vous ne pouvez pas créer de nouveaux fichiers dans ce dossier (qui est inclus dans le fichier apk du projet).

Si vous avez un fichier de base de données dans le dossier DirAssets, vous devez obligatoirement le copier dans un autre dossier sur l’appareil avant de pouvoir l’utiliser.  
  
**File.DirInternal / File.DirInternalCache**- Ces deux dossiers sont stockés dans la mémoire principale de l’appareil et sont privés pour l’application. Aucune autre application ne peut accéder aux fichiers dans ces dossiers.  
Le dossier ‘cache’ peut être effacé par le système d’exploitation s’il a besoin de plus d’espace mémoire.  
  
**File.DirRootExternal**Dossier racine de la carte mémoire interne.  
  
**File.DirDefaultExternal**- Le dossier par défaut de votre application sur la carte SD.  
Ce dossier est : <storage card>/Android/data/<package>/files/  
Il sera créé si nécessaire.

Notez que l’appel à une des deux propriétés ci-dessus va ajouter l’autorisation EXTERNAL\_STORAGE à votre application.  
  
Conseil : Vous pouvez vérifier s’il existe une carte mémoire et si elle est disponible avec **File.ExternalReadable** et **File.ExternalWritable**.

Pour vérifier si un fichier existe, utilisez :

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

Renvoie True si le fichier existe sinon False.

L’objet File contient plusieurs méthodes pour écrire ou lire dans des fichiers.

Pour pouvoir écrire ou lire dans un fichier il doit être ouvert.

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

- Ouvre le fichier spécifié pour écrire (Output), le paramètre Append () indique si le texte doit être ajouté à la fin du fichier existant ou non. Si le fichier n’existe pas, il sera créé.

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

- Ouvre le fichier pour lire (Input).

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

- Ecrit le texte (String) donné dans un nouveau fichier.

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

- Lit le fichier et renvoie son contenu dans un objet String.

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

- Écrit toutes les valeurs contenues dans un objet List dans le fichier. Toutes les valeurs sont converties dans des objets String si nécessaire. Chaque valeur est stockée dans une ligne séparée.  
Notez que, si une valeur contient un caractère ‘saut de ligne’ elle sera stockée sur plusieurs lignes, mais lors de la lecture, chaque ligne sera considérée comme un élément.

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

- Lit le fichier et stocké chaque ligne comme élément dans l’objet List.

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

**-** Le contenu de l’objet Map, qui contient des paires de valeurs (valeur clé et valeur élément), sera stocké dans un fichier texte. Le format du fichier est connu sous Java Properties: [.properties - Wikipedia](https://fr.wikipedia.org/wiki/.properties)  
Le format du fichier est sans importance tant qu’on ne l’édite pas manuellement. Ce format rend son édition plus facile.  
Une utilisation courante de File.WriteMap est le stockage, dans un fichier, d’un objet Map de ‘paramétrages’.

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

**-** Lit un fichier de propriétés et renvoie les paires de valeurs clé – valeur en objet Map. Notez que l’ordre des éléments peut être différent de l’ordre d’origine.

Quelques autres fonctions utiles :

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

- Copie le fichier source (FileSource) du dossier source (DirSource) dans le dossier de destination (DirTarget) sous le nom (FileTarget).

Notez qu’il est impossible de copier des fichiers dans le dossier DirAssets.

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

- Efface le fichier spécifié (FileName) du dossier spécifié (Dir).

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

- Liste les fichiers et sous-dossiers contenus dans le dossier (Dir) dans un objet List.

Example :

Private List1 As List

List1 = File.ListFiles(File.DirRootExternal)

List1 peut être déclaré dans Sub Globals

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

- Renvoie le nombre d’octets du fichier (FileName).

Cette fonction ne peut pas être utilisée pour des fichiers dans le dossier DirAssets.

### Noms de fichiers

Les noms de fichiers dans B4x autorisent les caractères suivants :

**a** à **z**, **A** à **Z**, **0** à **9**, point **.** souslignage **\_** plus les caractères suivants **+ - % &**

Des espaces et les caractères suivants **\* ?** ne sont pas autorisés.

Exemple : MonFichier.txt

Notez que les noms de fichier dans B4x sont sensibles à la casse !

MonFichier.txt est différent de monfichier.txt

### Sous-dossiers

Vous pouvez créer des sous-dossiers dans B4x avec.

File.MakeDir(File.DirInternal, "Images")

Pour accéder à ce dossier vous devez ajouter le nom du sous-dossier au nom du dossier avec "/" entre les deux.

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

Ou ajouter le nom du sous-dossier devant le nom du fichier avec "/" entre les deux.

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

Les deux possibilités fonctionnent.

### B4A, B4J Objet TextWriter / écriture de textes

Il y a deux autres objets utiles pour des fichiers texte : **TextWriter** et TextReader :

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

- Initialise un objet TextWriter en tant que flux d’écriture (output).

Exemple :

Private Writer As TextWriter

Writer.Initialize(File.OpenOutput(File.DirRootExternal, "Test.txt" , False))

Writer peut être déclaré dans la routine Sub Globals.

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

- Initialise un objet TextWriter en tant que flux d’écriture (output).

- *Encoding* indique le code de caractères (CharacterSet) pour l’encodage du texte (voir chapitre suivant).

Exemple :

Private Writer As TextWriter

Writer.Initialize2(File.OpenOutput(File.DirRootExternal,"Test.txt" ,False)," ISO-8859-1")

Writer peut être déclaré dans la routine Sub Globals.

Voir : [Encodage de texte](#_Text_encoding)

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

- Écrit le texte vers le flux.

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

- Écrit le texte vers le flux suivi du caractère saut de ligne LF ou Chr(10) ou CRLF.

**TextWriter.WriteList** (List As List)

- Écrit chaque élément de l’objet List dans une ligne.

Notez que si le contenu d’un élément contient le caractère saut de ligne, LF ou Chr(10) ou CRLF, il sera écrit sur deux lignes (lors de la lecture deux éléments seront lus).

Tous les contenus des éléments seront convertis en objets String.

**TextWriter.Close**

- Ferme le flux.

Exemple :

Private Writer As TextWriter

Writer.Initialize(File.OpenOutput(File.DirDefaultExternal, "Text.txt", False))

Writer.WriteLine("Ceci est la première ligne")

Writer.WriteLine("Ceci est la deuxième ligne ")

Writer.Close

### B4A, B4J Objet TextReader / lecture de textes

Il y a deux autres objets utiles pour des fichiers texte : TextWriter and **TextReader** :

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

- Initialise un objet TextReader en tant que flux de lecture (input).

Exemple :

Private Reader As TextReader

Reader.Initialize(File.OpenInput(File.DirRootExternal, "Test.txt"))

Reader peut être déclaré dans la routine Sub Globals.

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

- Initialise un objet TextReader en tant que flux de lecture (input).

- *Encoding* indique le code de caractères (CharacterSet) pour l’encodage du texte (voir chapitre suivant).

Exemple :

Private Reader As TextReader

Reader.Initialize2(File.OpenInput(File.DirRootExternal, "Test.txt", "ISO-8859-1")

Reader peut être déclaré dans la routine Sub Globals.

Voir : [Encodage de texte](#_Text_encoding)

**TextReader.ReadAll** As String

- Lit tout le texte restant et ferme le flux.

Exemple :

txt = Reader.ReadAll

**TextReader.ReadLine** As String

- Lit la ligne suivante.

Les caractères saut de ligne LF ne sont pas lus.

Renvoie Null s’il n’y a plus de caractères à lire.

Exemple :

Private Reader As TextReader

Reader.Initialize(File.OpenInput(File.DirDefaultExternal, "Text.txt"))

Private line As String

line = Reader.ReadLine

Do While line <> Null

Log(line)

line = Reader.ReadLine

Loop

Reader.Close

**TextReader.ReadList** As List

- Lit le texte restant et renvoie un objet List avec un élément par ligne.

Puis ferme le flux.

Exemple :

List1 = Reader.ReadList

### Encodage de texte

Encodage de texte ou codage de caractères en un code (aussi appelé page de code) qui associe un jeu de caractères abstraits d’un ou plusieurs systèmes d’écriture (comme des alphabets ou des syllabaires) utilisés pour transcrire des langues naturelles avec une représentation numérique pour chaque caractère de ce jeu, ce nombre pouvant lui-même avoir des représentations numériques différentes (source Wikipedia).

La page de code par défaut dans B4x est Unicode UTF-8.

Dans Windows les pages de code les plus utilisées sont ASCII et ANSI.

* ASCII contient les définitions pour 128 caractères, dont 33 caractères de contrôle non imprimables (aujourd’hui obsolète pour la plupart) que déterminent comment le texte est réparti dans l’espace.
* ANSI, Windows-1252 ou CP-1252 est une page de code pour les alphabets latins, utilisé par défaut dans les composants de Microsoft Windows en Anglais et d’autres langues occidentales avec 256 définitions. Les 128 premiers caractères sont les mêmes que pour la page de code ASCII.

Beaucoup de fichiers générés par des programmes Windows sont encodé avec la page de code ANSI dans les pays occidentaux. Par exemple : fichiers csv Excel, fichiers créés avec Notepad par défaut. Mais, on peut aussi enregistrer des fichiers avec l’encodage *UTF-8*.

B4x peut utiliser les pages de code suivants :

* UTF-8 page de code par défaut
* UTF -16
* UTF - 16 BE
* UTF - LE
* US-ASCII page de code ASCII
* ISO-8859-1 pratiquement équivalent à ANSI
* Windows-1251 page de code pour caractères cyrilliques
* Windows-1252 page de code pour alphabets latins

Pour lire des fichiers encodés avec ANSI utilisez la page de code *Windows-1252*.

Si vous devez enregistrer des fichiers qui doivent aussi être utilisé sous Windows, utilisez également la page de code *Windows-1252*.

Une autre différence entre B4x et Windows est le caractère fin de ligne :

* B4x, seul le caractère saut de ligne LF (Line Feed) caractère Chr(10) est ajouté à la fin d’une ligne.
* Windows, deux caractères CR (retour chariot Chr(13)) et LF Chr(10) sont ajoutés à la fin d’une ligne. Si vous devez enregistrer des fichiers pour Windows vous devez ajouter le caractère CR vous même.

Les symboles de fin de ligne sont :

* B4x CRLF Chr(10)
* Basic4PPC CRLF Chr(13) & Chr(10)

Pour lire ou écrire des fichiers avec un encodage différent vous devez utiliser les objets TextReader ou TextWriter avec les méthodes d’initialisation Initialize2. Même pour des fichiers csv.

Conseil pour lire des fichiers csv Excel :

Vous pouvez soit :

* Sur le PC, lire le fichier csv dans un éditeur de texte comme *Notepad++*
* Enregistrer le fichier avec l’encodage *UTF-8*.  
  Avec *Notepad++* utilisez le code Encode in UTF-8 without BOM, voir ci-dessous.

Ou

* Lire le fichier avec TextReader.Initialize2 et l’encodage "Windows-1252".
* Le réenregistrer avec TextWriter.Initialize avec l’encodage standard de B4x.
* Lire le fichier avec LoadCSV ou LoadCSV2 de la bibliothèque StringUtils.

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)

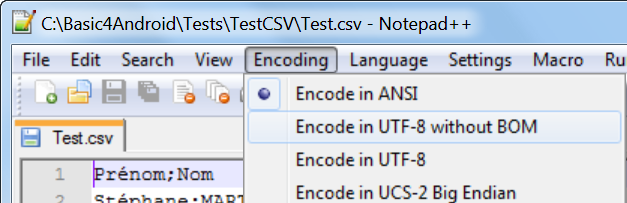
Si vous enregistrez un fichier avec NotePad et l’encodage UTF-8, trois caractères supplémentaires sont ajoutés au début du fichier.

Ces octets sont appelés **indicateur d'ordre des octets** ou BOM en anglais pour (Byte Oder Mark).

Pour *UTF-8* ils sont définis par cette séquence d’octets : 0xEF,0xBB,0xBF.

Un éditeur de texte ou un navigateur web qui interprète ce texte comme étant un texte *Windows-1252* va afficher ces caractères ï»¿.

Pour éviter ces caractères vous pouvez utiliser *Notepad++* au lieu de *NotePad* et utiliser *Encode in UTF-8 without BOM*.



Une autre possibilité pour changer un texte d’un encodage *Windows-1252* vers *UTF-8* est d’utiliser le code ci-dessous.

Private var, result As String

var = "Gestió"

Private arrByte() As Byte

arrByte = var.GetBytes("Windows-1252")

result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")

## Objet List / Liste B4A, B4i et B4J seulement

Les objets List sont similaires à des tableaux dynamiques.

Un objet List doit être initialisé avant de pouvoir l’utiliser.

* Initialize Initialise un objet List vide.  
  Private List1 As List  
  List1.Initialize  
  List1.AddAll(Array As Int(1, 2, 3, 4, 5))
* Initialize2 (SomeArray)  
  Initialise un objet List avec les valeurs données dans le tableau *SomeArray*. Cette méthode peut être utilisé pour convertir des tableaux (arrays) en un objet List. Notez que si vous passez un objet List au lieu d’un tableau, alors les deux objets partagent les mêmes données, et si vous passez un tableau, la liste aura une longueur fixe.   
  Ce qui veut dire que vous ne pourrez pas ajouter ni supprimer des éléments.  
  Exemple 1:  
  Private List1 As List  
  List1.Initialize2(Array As Int(1, 2, 3, 4, 5))  
  Example 2:  
  Private List1 As List  
  Private SomeArray(10) As String  
  ' Remplir le rableau ici  
  List1.Initialize2(SomeArray)

Vous pouvez ajouter ou supprimer des éléments d’un objet List, la taille sera changée en conséquence.

Avec :

* Add (item As Object)   
  Ajoute un élément à la fin de l’objet List.  
  List1.Add(Value)
* AddAll (Array As String("value1", "value2"))  
  Ajoute tous les éléments de tableau à la fin de l’objet List.  
  List1.AddAll(List2)  
  List1.AddAll(Array As Int(1, 2, 3, 4, 5))
* AddAllAt (Index As Int, List As List)  
  Insère tous les éléments du tableau à partir de index spécifié.  
  List1.AddAll(12, List2)  
  List1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))
* InsertAt (Index As Int, Item As Object)  
  Insère l’élément à l’index spécifié.   
  Tous les éléments au-dessus sont décalés d’un rang vers le haut.  
  List1.InsertAt(12, Value)
* RemoveAt (Index As Int)  
  Supprime l’élément à l’index spécifié.  
  List1.RemoveAt(12)

Un objet List peut contenir n’importe quel type d’objet. Dans B4A, si un objet List est déclaré dans la routine Sub Process\_Globals in ne peut pas contenir des objets d’Activity (comme des views).  
B4x convertit automatiquement des tableaux en objet List. Si un objet List est demandé comme paramètre on peut sans autre passer un tableau.

Taille d’un objet List :

* List1.Size

Utilisez la méthode Get pour extraire un élément de l’objet List (le premier index est 0).

Pour extraire le premier élément utilisez Get(0).

Pour extraire le dernier élément utilisez Get(List1.Size - 1).

* Get(Index As Int)  
  nombre = List1.Get(i)  
    
  You can use a For loop to iterate over all the values:  
  For i = 0 To List1.Size - 1  
   Private nombre As Int  
   nombre = List1.Get(i)  
   ...  
  Next

Des objets List peuvent être enregistrés et lus dans des fichiers avec :

* 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")

La valeur d’un élément peut être modifié avec :

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

Un objet List peut être trié (tous les éléments doivent obligatoirement être des nombres ou des objets String) avec :

* Sort(Ascending As Boolean)  
  List1.Sort(True) tri ascendant.  
  List1.Sort(False) tri descendant.
* SortCaseInsensitive(Ascending As Boolean) tri indépendant de la casse.

Vider un objet List avec :

* List1.Clear

## Objet Map B4A, B4i et B4J seulement

Un objet Map est une collection de paires de valeurs, une valeur clé et une valeur qui lui est associée.

Les clés sont uniques. Ce qui veut dire que si vous ajoutez une nouvelle paire clé / valeur et qu’il existe déjà une même clé, la valeur précédente sera remplacée par la nouvelle.

La clé doit être un objet String ou un nombre. La valeur associée peut être n’importe quel objet.

Similaire à l’objet List, un objet Map peut contenir n’importe quel objet pour la valeur. Mais, dans B4x, s’il est déclaré dans la routine Process\_Globals il ne peut pas contenir des objets d’Activity (Views).

Les objets Map sont très utiles pour mémoriser des paramétrages d’applications.

Des objets Map sont utilisés dans cet exemple :

* Module DBUtils   
  utilisé pour des enregistrements de bases de données, les clés sont les noms des colonnes et les valeurs sont les valeurs dans la colonne.

Les objets Map doivent être initialisés avant de pouvoir les utiliser.

* Initialize Initialise un objet Map vide.  
  Private Map1 As Map  
  Map1.Initialize

Ajouter un nouvel élément :

* Put(Key As Object, Value As Object)  
  Map1.Put("Langue", "English")

Extraire un élément :

* Get(Key As Object)  
  Language = Map1.Get("Langue")

Extraire la clé ou la valeur à l’index défini (seulement B4A et B4J) :

Renvoie la valeur de l’élément à l’index défini.

GetKeyAt et GetValueAt ne devraient être utilisés que pour itérer à travers tous les éléments.

Ces méthodes sont optimisées pour itérer dans le sens ascendant.

* GetKeyAt(Index As Int)  
  Key = Map1.GetKeyAt(12)

Extraire la valeur à l’index défini (seulement B4A et B4J) :

* GetValueAt(Index As Int)  
  Value = Map1.GetValueAt(12)

Vérifier si l’objet Map contient déjà une entrée avec la donnée :

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

Supprimer un élément :

* Remove(Key As Object)  
  Map1.Remove("Langue")

Vider l’objet Map, supprime tous les éléments :

* Clear  
  Map1.Clear

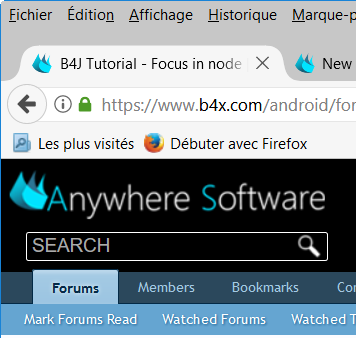
Les objets Map peuvent être enregistrés et lus dans des fichiers avec :

* 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)  
  Lit le fichier et décompose chaque ligne en une paire clé - valeur (en objets String).  
  Notez que l’ordre des éléments dans l’objet Map peut être différent que dans le fichier.  
  mapParametres = File.ReadMap(File.DirInternal, "Parametres.txt")
* File.ReadMap2(Dir As String, FileName As String, Map As Map)  
  Similaire à ReadMap. ReadMap2 ajoute les éléments à l’objet Map défini.  
  En utilisant ReadMap2 avec un objet Map contenant déjà des éléments vous pouvez forcer l’ordre des éléments à vos besoins.  
  mapParametres = File.ReadMap2(File.DirInternal, "Parametres1.txt", mapParametres)

# Outils d’aide

Les outils suivants sont utiles pour trouver des réponses à beaucoup de vos questions.

## Fonction recherche dans le forum / Search



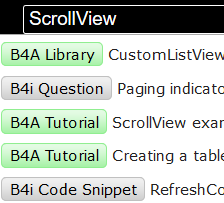
Dans le coin supérieur gauche vous trouvez le champ de recherche ‘Search’ pour le forum. Selon la largeur de la fenêtre, le champ ‘Search’ peut se trouver dans le coin supérieur droit.

Entrez une question ou un mot clé puis presseur ‘Entrée’.

La fonction montre les posts qui correspondent à votre requête.



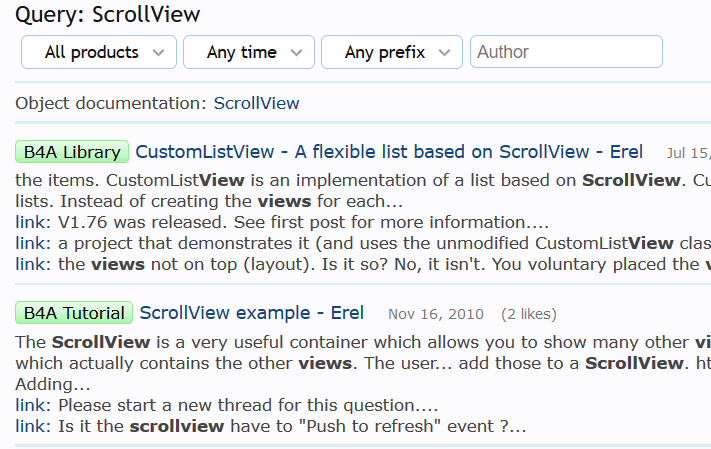
Exemple : Entrez le mot clé ScrollView.



Une liste de résultats est affichée juste en dessous du champ de recherche.

Cliquez sur un élément de la liste pour afficher le ‘post’ complet.

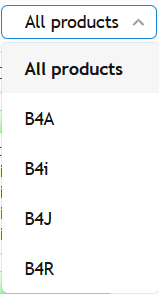
Et le résultat :



Sur le haut vous trouvez des boutons permettant un filtrage selon différents critères.

Cliquz sur le titre pour afficher le ‘post’ complet.

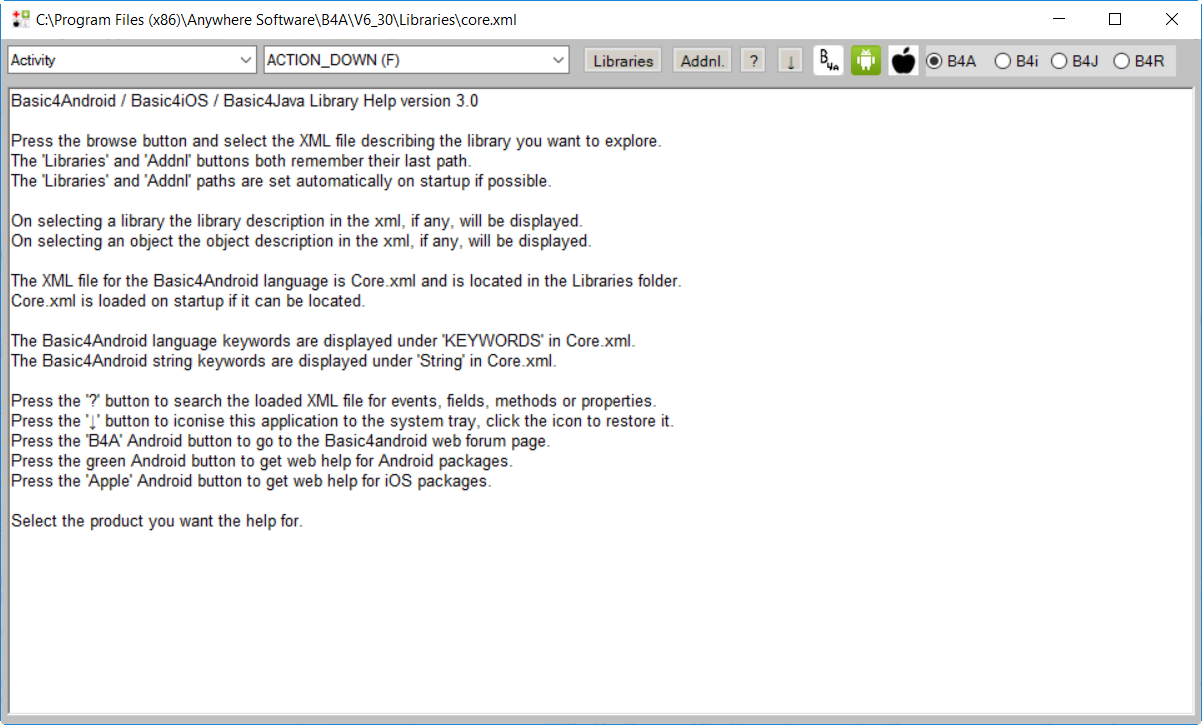
Exemple de filtrage sur un produit :



## B4x Help Viewer

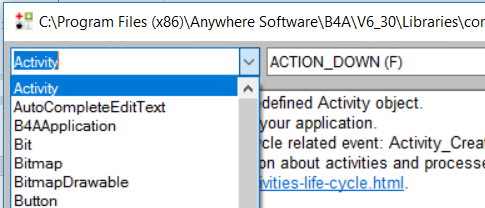
Ce programme affiche les fichiers d’aide xml. Il était écrit à l’origine par Andrew Graham (agraham) pour B4A. Je l’ai modifié, avec l’agrément d’Andrew, pour afficher les fichier xml de toutes les plateformes B4A, B4J, B4i et B4R.

Le programme peut être [téléchargé](http://www.basic4ppc.com/android/forum/threads/b4i-help-viewer.46969/) depuis le forum.

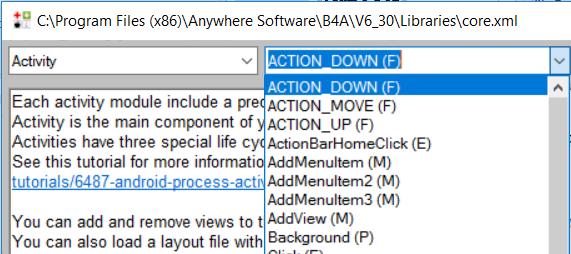


Sur le haut, on trouve :





Dans le coin supérieur gauche se trouve une liste déroulante qui affiche les différents objets inclus dans la bibliothèque sélectionnée.



A côté de la liste des objets vous trouvez une autre liste déroulante avec

* méthodes(M)
* événements(E)
* propriétés(P)
* champs(F)

pour l’objet sélectionné.

 Sélectionne les bibliothèques standards (celles livrées avec B4x).

 Sélectionne les bibliothèques additionnelles.

 Moteur de recherche pour trouver des objets avec des mots clé.

 Ferme B4AHelp

 Lance le forum 'Online Community'.

 Lance le site Android Developers.

 Lance le site iOS Developers.

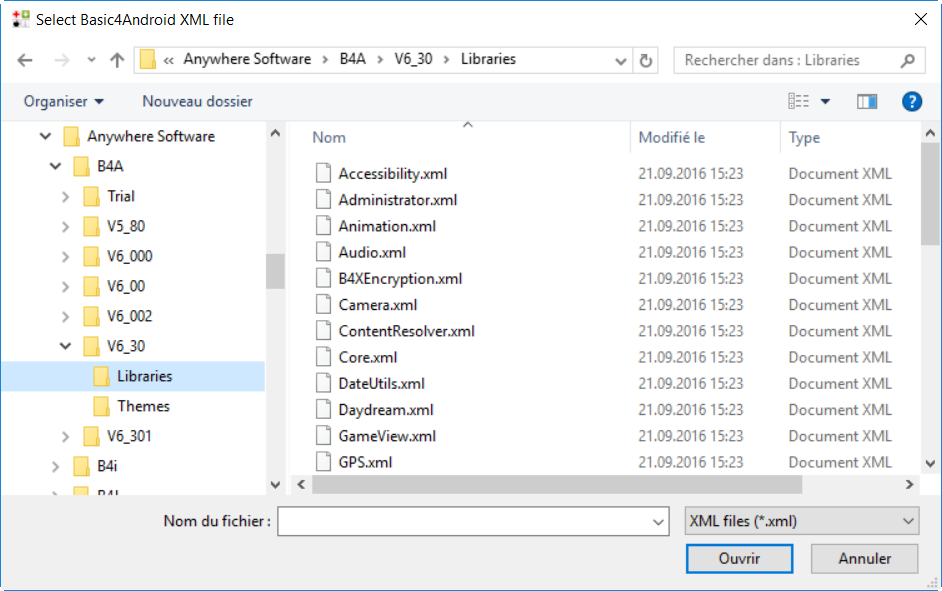
 Fichiers d’aide B4A.

 Fichiers d’aide B4i.

 Fichiers d’aide B4J.

 Fichiers d’aide B4R.

 Bibliothèques standard.



Sélectionnez la bibliothèque et cliquez sur  .

Ici  vous pouvez sélectionner le dossier dans lequel les bibliothèques standard sont sauvées.

Une fois sélectionné, le nom du dossier est mémorisé pour les démarrages suivants du programme.

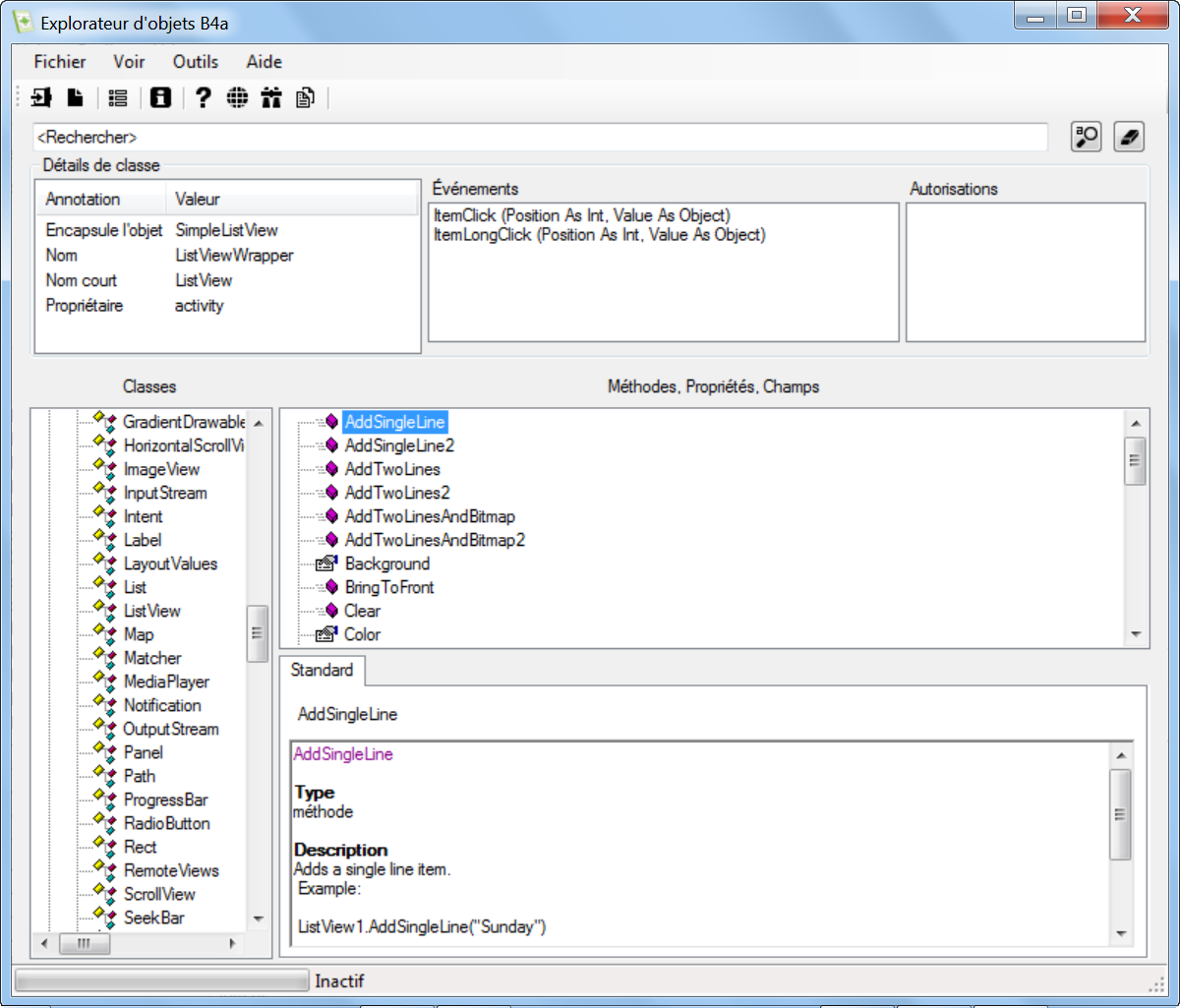
## 

## Help documentation - B4A Object Browser

Ceci est aussi un programme autonome Windows affichant les fichiers d’aide des bibliothèques.

Il a été écrit par Vader et peut être téléchargé [ici](http://www.basic4ppc.com/forum/basic4android-getting-started-tutorials/25682-help-documentation-docloader-16.html#post148746).

Un mode d’emploi en pdf (en anglais) sur son utilisation fait partie du téléchargement.



# Conseils

Quelques conseils d’Erel pour les développeurs B4x ([[B4X] Tips for B4X developers](https://www.b4x.com/android/forum/threads/b4x-tips-for-b4x-developers.62121/#post-510240)).

## Séparer les données du code

Introduire des données directement dans le code rend votre programme illisible et difficile à maintenir.   
Il y a plusieurs possibilités de traiter les données. Par exemple, vous pouvez ajouter un fichier texte dans l’onglet Gestionnaire de fichiers et le lire dans une liste avec :

Dim data As List = File.ReadList(File.DirAssets, "SomeFile.txt")

## Ne vous répétez pas

Ne copiez pas plusieurs fois une même portion de code, car si plus tard vous devez faire une modification vous devez la faire x fois, il vaut alors la peine de réfléchir à une solution plus élégante, définr une routine par exemple.   
Du code répétitif est difficile à maintenir et mettre à jour. Le mot clé Sender peut être utile dans bien des cas (tutoriel ancien, mais toujours d’actualité : [Tick-Tack-Toe: working Wit aras of views](https://www.b4x.com/android/forum/threads/8506/#content)).

## Collection Map

Tous les développeurs devraient apprendre comment utiliser une collection Map. C’est de loin la collection la plus utile. Tutoriel : <https://www.b4x.com/android/forum/threads/map-collection-the-most-useful-collection.60304/>

## Nouvelles technologies et fonctionnalités

Ne soyez pas effrayé d’apprendre de nouvelles choses. En tant que développeurs nous sommes obligés d’apprendre du nouveau. Tout évolue, que nous le voulions ou non. La technologie [MQTT](https://www.b4x.com/android/forum/threads/59471/#content) est un bon exemple. Cette technologie ne m’étais pas familière, mais lorsque j’ai commencé par l’apprendre j’ai été surpris de la puissance et de la facilité de cette solution.  
Vous devriez prendre en considération les fonctionnalités ci-dessous:  
- Smart strings: <https://www.b4x.com/android/forum/threads/50135/#content>  
- Itérateur For Each: <https://www.b4x.com/android/forum/threads/loops.57877/>  
- Classes: <https://www.b4x.com/android/forum/threads/18626/#content>

## Logs / affichage

Vous devriez afficher des commentaires (Logs) durant le déroulement du programme. Surtout s’il y a des erreurs. Si vous ne voyez pas ces commentaires pour une raison ou une autre vous devez prendre le temps pour trouver la raison et corriger le problème. Avec B4A-Bridge les commentaires ne seront affichés qu’en mode Debug. Si vous rencontrez un problème qui n’apparait qu’en mode Release, vous devez connecter votre dispositif au moyen d’un cable USB.

## B4A, Evitez les appel de DoEvents

DoEvents interfère avec la queue interne de messages et peut provoquer des problems inattendus. Il y a très peu de cas où c’est nécessaire. Ce n’était pas le cas lorsque la version 1.0 de B4A a été publiée. Depuis, les bibliothèques se sont améliorées et offrent maintenant de meilleures solutions. Par exemple, si des operations de bases de données sont trop longues, tout en utilisant correctement les transactions, vous devriez vous tourner vers les méthodes asynchrones. Ou utiliser [Sleep](#_Sleep) ou [Wait For](#_Wait_For).

## L’objet Strings contient des caractères et non des octets

N’essayez pas de sauver des octets (bytes) en tant qu’objet String. Ca ne fonctionne pas. Utilisez un tableau d’octets à la place. La manière correcte de convertir des octets en un objet String est un encodage base 64 ou avec ByteConverter.HexFromBytes.

## B4A Utilisez des services, spécialement le service Starter

Les Services sont plus simples que les Activities. Ils ne sont jamais desactivés (paused) et sont pratiquement toujours accessibles.   
**Trois règles générales à propos des variables globales:**  
1. Toutes les variables non IHM (interface homme-machine) doivent être déclarées dans Process\_Globals.   
2. Les variables publiques (process\_global) devraient être déclarées dans le Service Starter dans Process\_Globals et la definition de leur valeurs dans la routine Service\_Create.  
3. Les variables process globales dans les Activities doivent être définies seulement lorsque FirstTime est égal à True.  
  
Ceci ne concerne que B4A. C’est plus simple dans B4J et B4i car il n’y a pas de cycles de vie particuliers et les modules ne sont jamais desactivés.

## Layouts IHM

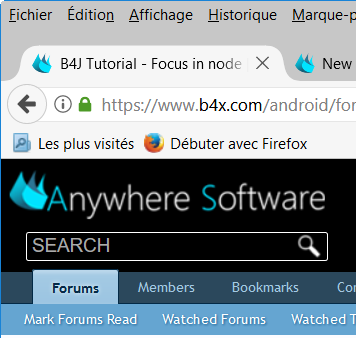
B4X fournit quelques outils qui vous aident à définir des layouts flexibles qui s’adaptent à toutes les grandeurs d’écran. Les outils principaux sont: ancres (anchors) et les Scripts dans le Concepteur visual (Designer Scipts). Évitez d’ajouter des variantes (variants) multiples (deux sont bien). Les variantes ont été introduites dans la version 1.00 de B4A avant l’introduction des autres fonctionnalités. Les variantes sont difficiles à maitenir et peuvent être remplacées par des scripts.  
Les ancres (anchors) sont simples à utiliser et très puissants.  
N’abusez pas des unites pourcentage (%x ou %y), à moins que vous ne programmiez des jeux.  
[http://www.basic4ppc.com/forum/basi...ing-multiple-screens-tips-best-practices.html](http://www.basic4ppc.com/forum/basic4android-getting-started-tutorials/17647-supporting-multiple-screens-tips-best-practices.html)

## B4J une solution en arrière plan

B4A, B4i, B4J partagent le même language, les même concepts et les mêmes APIs. Il est facile d’échanger des données entre les différentes plateformes avec B4XSerializator.   
Il est facile d’implémenter des solutions de serveur puissantes avec B4J. Spécialement si les clients sont développés avec B4A, B4i ou B4J.

## Recherche (Search)

Utilisez la [function recherche](#_Fonction_recherche_dans) (Search) dans le forum. Vous pouvez filtrer les résultats en ajoutant la plateforme (par exemple b4a) à la demande ou en cliquant sur un des boutons dans la page de résultats.  
La plupart des questions trouvent leur réponse en quelques clics.



## Notepad++.

A un moment ou un autre, nous devons travailler avec des fichiers texte. Je vous recommande vivement d’utiliser l’éditeur Notepad++ qui permet d’afficher l’encodage, les caractères de fin de ligne et inclut d’autres fonctionnalités. <https://notepad-plus-plus.org/>

# Dictionnaire

* **Activity** Dans B4A objet activité, équivalent à un écran.
* **EDI E**nvironment de **D**eveloppment **I**ntégré, l’éditeur de B4x.   
   IDE en anglais, Integrated Development Environment.
* **IHM I**nterface **H**omme – **M**achine  
   UI en anglais, User Interface
* **Label** Objet d’interface homme-machine: étiquette. Visualise du texte.
* **Layout** Terme anglais significant ‘mise en page’, utilisé pour les interfaces homme  
   machine.
* **Node** Dans B4J, objet d’interface homme-machine générique.
* **Panel** Dans B4A et B4i, objet d’interface homme-machine: panneau.
* **Pane** Dans B4J, objet d’interface homme-machine: panneau.
* **View** Dans B4A et B4i, objet d’interface homme-machine générique.