Automatiser Excel et Numbers pour Mac avec RealBasic 2009 et AppleScript

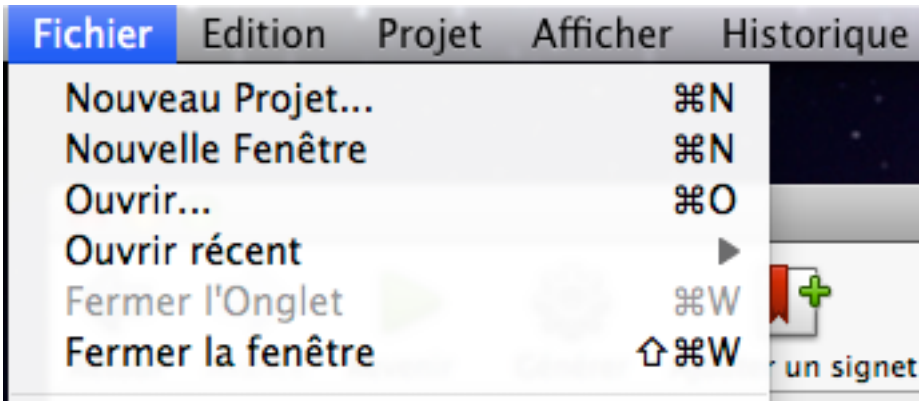
I Introduction

Ce tutoriel va vous montrer comment automatiser des applications comme Excel ou Numbers sous Mac OS X, en utilisant AppleScript et le langage de programmation RealBasic. Vous apprendrez à insérer des valeurs dans des cellules de tableaux, à paramétrer des tableaux. Je ferai un parallèle avec les technologies Windows. En effet, sous Windows dès que l'on veut automatiser des applications Office on utilise soit Visual Basic pour Applications, soit OLE au sein de programme. Sous Mac OS X, ces technologies n'existe pas, nous avons recours à des technologies comme AppleScript, dont l'utilisation est assez simple quand on en connaît le fonctionnement. Dans ce tutoriel nous allons demander à l'utilisateur par l'intermédaire d'une petite interface graphique de saisir le nombre de colonnes et le nombre de lignes par colonnes qui serviront à créer le tableau Excel ou Numbers. Ensuite le programme se chargera de générer automatiquement des nombres aléatoirement et les enverra dans les cellules du tableau généré.

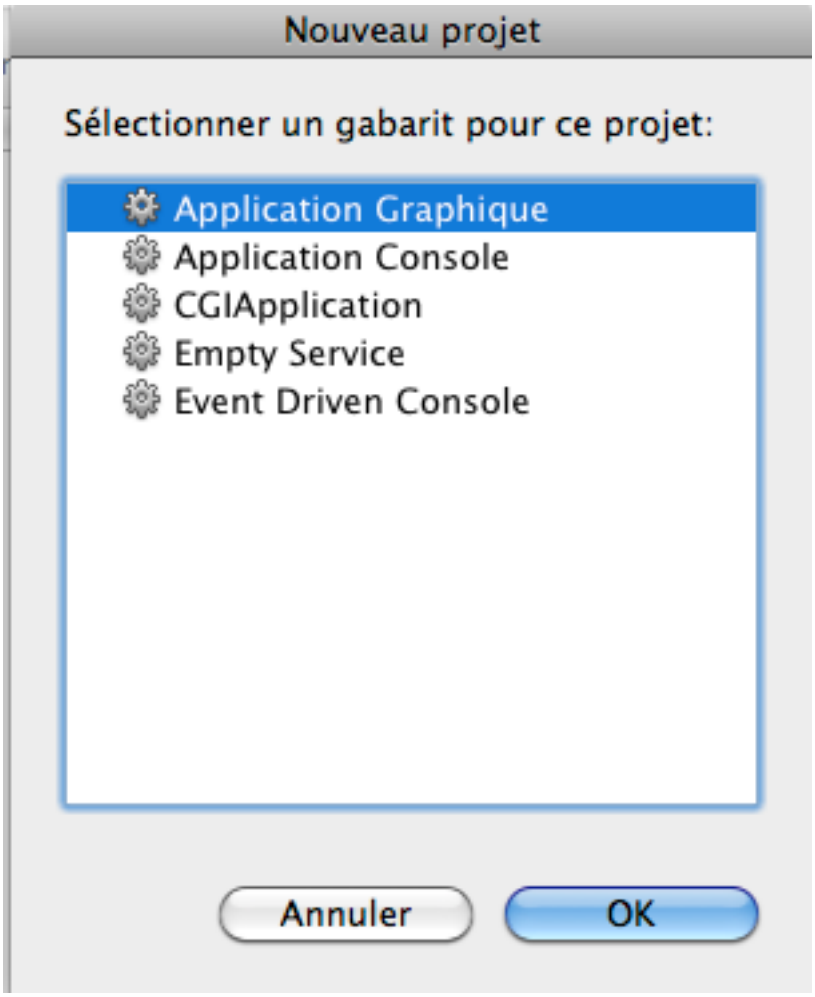
II Configuration du projet sous RealBasic 2009

1. Création du projet

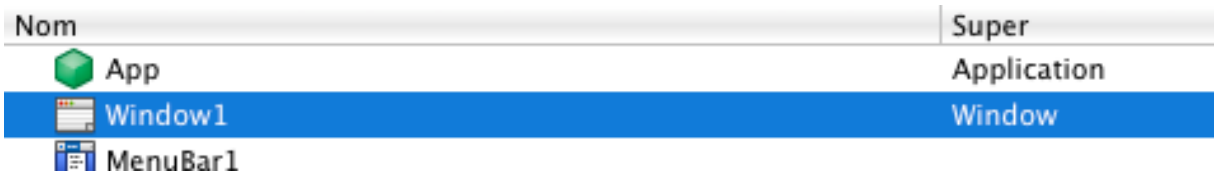
La première chose à réaliser dans l'immédiat est de créer le projet dans RealBasic 2009. Pour ce faire, il suffit de cliquer sur le menu Fichier et sélectionner la commande Nouveau Projet (ou appuyer sur cmd-N) comme ceci :



Une fois cette commande sélectionée, une nouvelle fenêtre s'ouvre vous demandant quell type de projet voulez-vous créer. Vous avez le choix entre une application avec une interface graphique, une application sans interface (application console), une application CGI, ou un Service Web. Voici un apreçu de cette fenêtre :



Pour ce tutoriel nous aurons besoin d'une interface graphique donc sélectionnez le type Application graphique.
L'environnement de programmation RealBasic 2009 génère le squelette du projet. Sur la fenêtre principale du projet, vous devez voir apparaître deux onglets. L'onglet Projet, qui va vous permettre d'administrer tous les éléments de votre projet.
Voici un apercu de ce que l'on peut observer sur cet onglet :



Vous pouvez voir dans la zone réservée aux fichiers que l'environnement a généré trois fichiers :

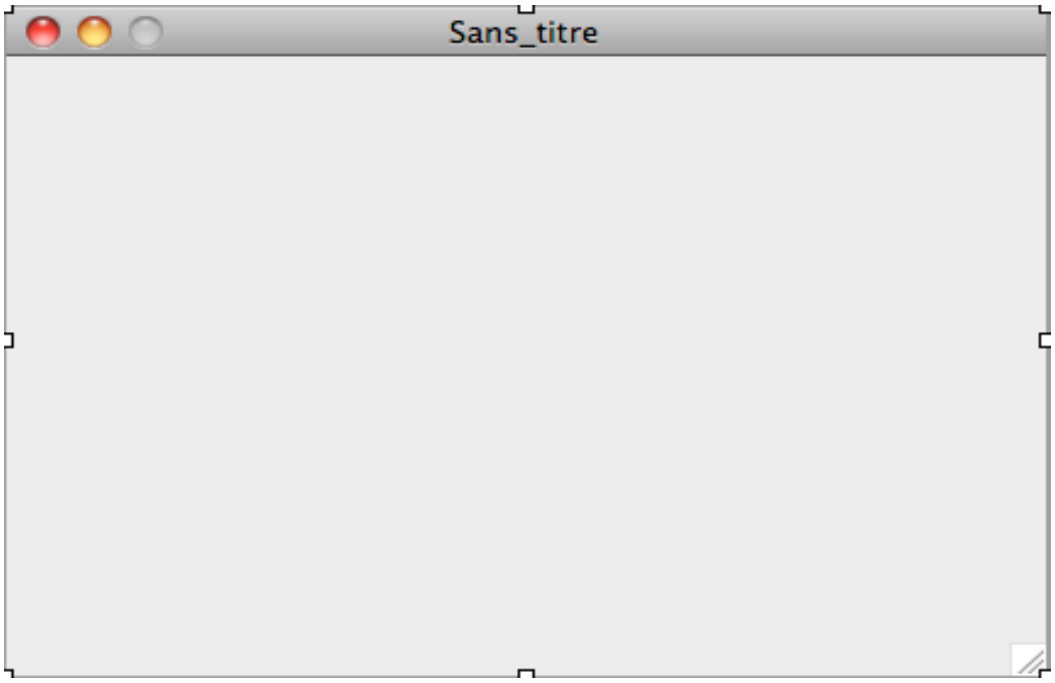
- App qui est le fichier exécutable de l'application. Le nom App n'est pas trop parlant nous verrons comment personnaliser ce nom plus tard.
- Window1 représente l'interface graphique de la fenêtre principale de l'application.
- MenuBar1 représente la barre de menu qui apparaît à l'exécution de l'application.

C'est par l'intermédiaire de cet onglet , que nous allons créer les fichiers annexes du projet comme les nouveaux modules ou encore un fichier de classe ou encore ajouter des resources à notre application.

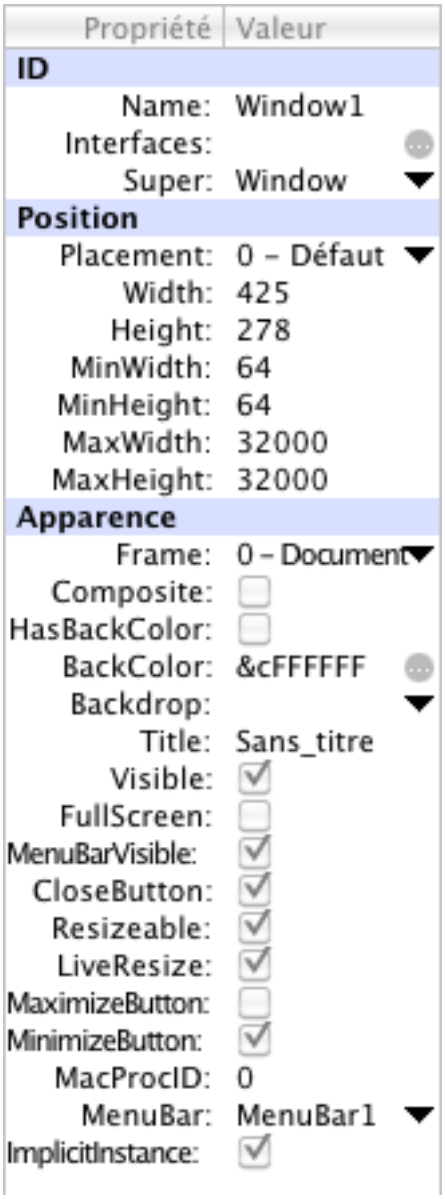
2. Création de la fenêtre principale

La fenêtre de l'application devrait ressembler grosso modo à ceci pour ce tutoriel:

Pour créer la fenêtre de l'application, il vous suffit juste de cliquer sur l'onglet Window1.
Nous allons donc personnaliser notre fenêtre.
Voici comment apparaît votre fenêtre à la creation de votre projet :



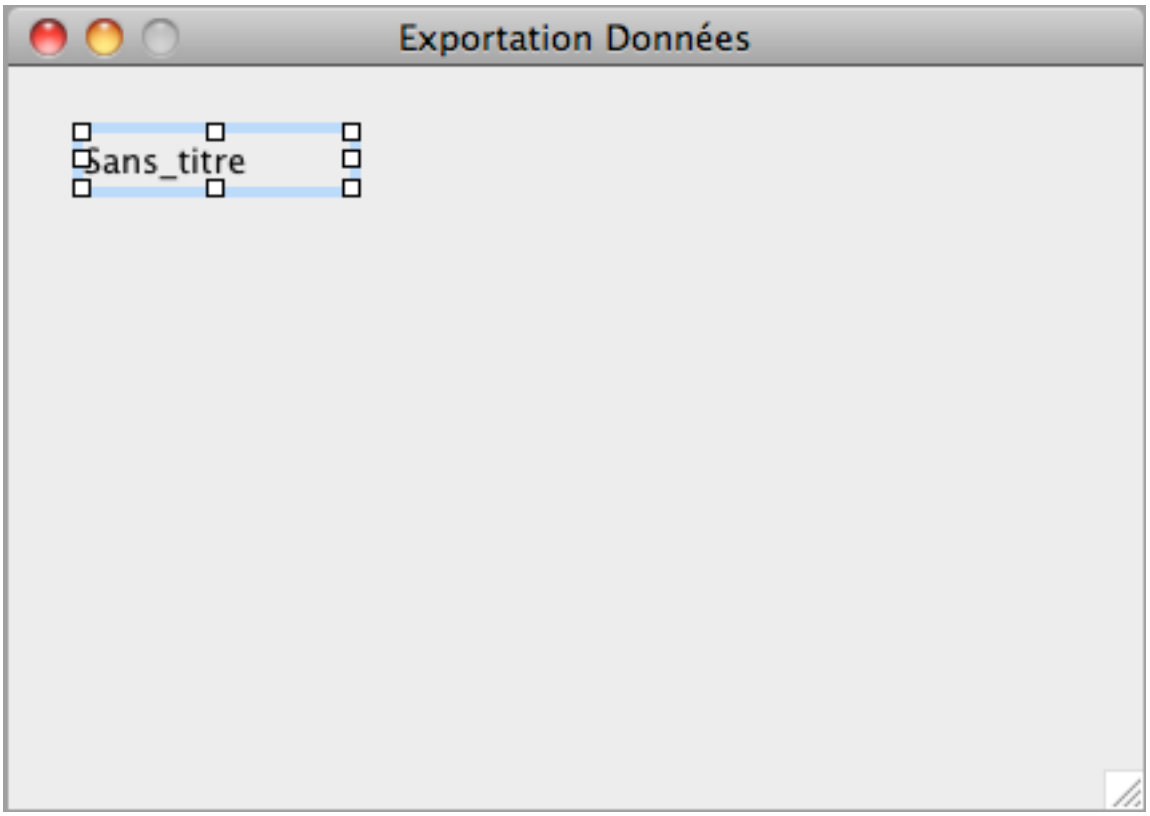
Rien que le titre affiché en haut sur la barre n'est pas très explicite il vous est possible de le personnaliser grâce à la fenêtre de propriétés suivante qui se trouve en général à droite sous l'interface graphique de RealBasic :



Pour la propriété Name changez la valeur correspondante par mainForm par exemple. Et pour la propiété Title donnez un nom plus explicite comme "Exportation données" par exemple.

Maintenant revenons en à notre fenêtre principale . Nous devons ajouter des composants à notre fenêtre pour que le résultat obtenu soit proche de celui que nous avons vu plus haut.
La fenêtre des composants se trouve à gauche dans l'interface graphique de Real Basic, il vous suffit d'effectuer un drag and drop du composant souhaité et de le déposer sur la fiche principale.

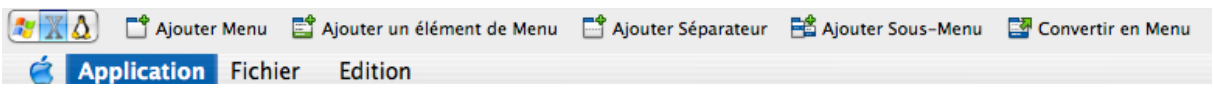
Par exemple dans le cas du composant StaticText (Label sous Visual Studio par exemple). Voici comment celui-ci apparaît sur la fiche après cette opération :



Le texte qui apparaît sur la fiche n'est pas très explicite non plus, donc comme pour la fiche principale vous allez modifier les propriétés Name et Text dans la fenêtre des propriétés associée.
La marche à suivre est identique pour tous les autres composants qui composent la fiche principale. A noter qu'au fur et mesure que vous deposez des composants sur la fiche principal des marques apparaissent pour vous aider à aligner ceci, comme le font des EDI comme Visual Studio ou Delphi sous la plateforme Windows.
Maintenant que la fiche principale correspond à peu près au résultat voulu, nous allons nous passer à la configuration du menu principal de l'application.

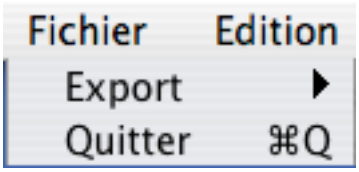
3. Personnalisation du menu de l'application

Nous allons maintenant personnaliser la barre de menu principale de l'application qui apparaît en haut lorsque vous lancez :

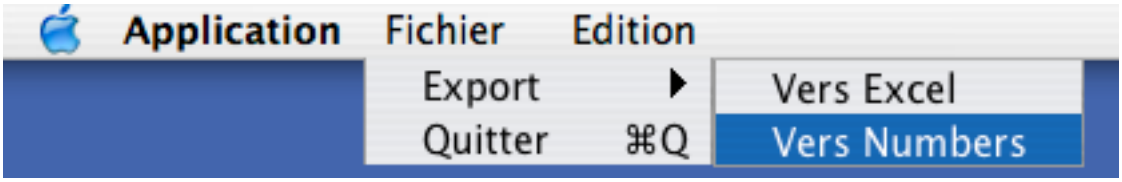


C'est le menu principal qui est généré par défaut lors de la création de l'application.

Vous avez en haut différentes commandes vous permettant facilement de personnaliser votre menu. Ce que nous voulons ajouter, ce sont deux commandes nous permettant d'exporter les données vers Excel ou Numbers.
Pour ce faire positionnez-vous sur la commande fichier et sélectionnez la commande "Ajouter Sous-Menu". Vous obtenez un nouvel élément de menu sous l'élément "Quitter" il vous suffit de le déplacer au dessus de l'élément Quitter si ceci n'est déjà fait… Voici le résultat obtenu:



Vous pouvez maintenant personnaliser le nom de la nouvelle commande. Dans la fenêtre des propriétés changez le texte de l'élément par "Export" et le nom par "menuExport".
Nous allons maintenant créer de sous menu de la même façon mais en sélectionnant la commande "Ajouter un élément de Menu"
Nous devrions obtenir comme résultat final ceci :



Vous pouvez de même ajouter un séparateur de menu entre Export et Quitter comme bon vous semble…
Vous avez donc vu comment personnaliser votre application grâce aux nombreux outils fournis par Real Basic. Les utilisateurs de Visual Studio ou autres EDI RAD sous Windows ne seront pas trop dépaysés et l'apprentissage des outils de Real Basic est somme toute assez aisée…
Maintenant que nous en avons fini avec notre interface graphique, nous allons passez à la partie codage de l'application.

III Codage de l'application

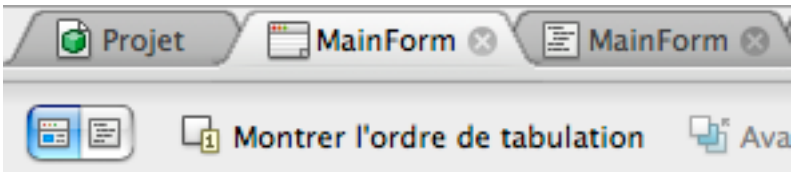
1. Introduction

Le but de l'application va être de générer une série de nombre aléatoire dans un tableau de un ou plusieurs dimensions et d'exporter ces données soit Excel, soit vers Numbers.
Nous allons dans un premier temps voir comment il est possible de déclarer des variables sous Real Basic.

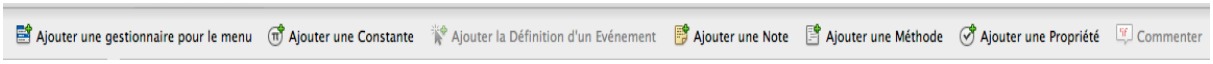
2. Accès à la fenêtre de codage de l'application

Tout d'abord pour accéder à la partie codage de votre application il faut double-cliquer sur le nom de votre fiche dans l'explorateur de projet (element associé au type

Window). Vous pouvez ensuite passer à la partie codage de votre application par l'intermédiaire du sélecteur suivant qui se trouve en haut à gauche de la fenêtre.



Vous accédez à la page associée vous permettant de coder différents éléments de votre application. Le principe de Real Basic est sensiblement différent en ce sens qu'il ne donne pas un apercu global du code mais vous avez accès à certaines parties du code par l'intermédiaire d'onglet (événements associés à des objets, propriétés, etc…).
Dans la partie gauche de la fenêtre vous avez la liste de tous les elements créé dans la fenêtre associée au design de la fiche ainsi que toutes les propriétés, methodes que vous avez créé… Il suffit de cliquer sur un élément pour éditer le code associé.
Dans la partie haute vous avez une barre d'outils vous permettant d'ajouter des éléments de codes à votre application.



Nous allons maintenant créer les propriétés qui seront utiles à notre application.

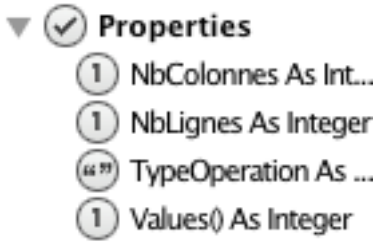
3. Déclaration des propriétés

Avant de pouvoir coder notre application nous allons declarer les quelques propriétés qui nous serviront pour la suite.
Pour ajouter une propriété à notre application il suffit de cliquer sur sur le bouton "Ajouter une Propriété " de la barre d'outils.
Un assistant s'ouvre pour créer votre propriété :

Vous pouvez remarquer déjà que la syntaxe est similaire à celle utilisée pour Visual Basic.
Vous allez donc créer ces quatre propriétés :

- NbColonnes de type Integer et initioalisez la à 0.
- NbLignes de type Integer et Initialisez la à 0.
- TypeOperation de type String pas d'initialisation.
- Un tableau Values (Values()) de type Integer pas d'initialisation.

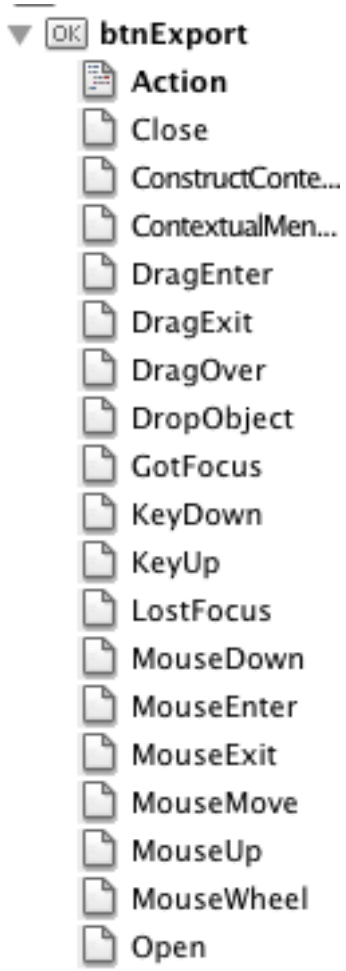
Vous pouvez alors remarquer que les propriétés créées apparaîssent bien dans la partie gauche de la fenêtre de codage dans la section Propriétés.



Il est possible de rééditer les propriétés comme bon vous semble tout au long de votre projet et d'en ajouter de nouvelles de la meme façon, un point positif pour le débutant.

4. Accès à l'éditeur de texte des événements liés aux composants

A ce niveau, les utilisateurs d'utres outils RADs sur PC ne seront pas dépaysés, en effet la plupart du temps il est possible de double-cliquer sur un composant sur la fenêtre de design pour activer le codage de l'événement principal qui lui correspond. Par exemple, si vous double-cliquez sur un bouton la fenêtre de code liée à l'événement Action (Click en général sur la plateforme Windows) de celui-ci sera activée...
Sinon vous avez accès à tous les événements associé à un composant donné sur la fenêtre réservé au codage par exemple pour un bouton nous avons :



Il suffit de cliquer sur un des événements pour ouvrir l'éditeur de code pour celui-ci...
Nous verrons plus tard plus en details le codage des différentes parties de l'application.

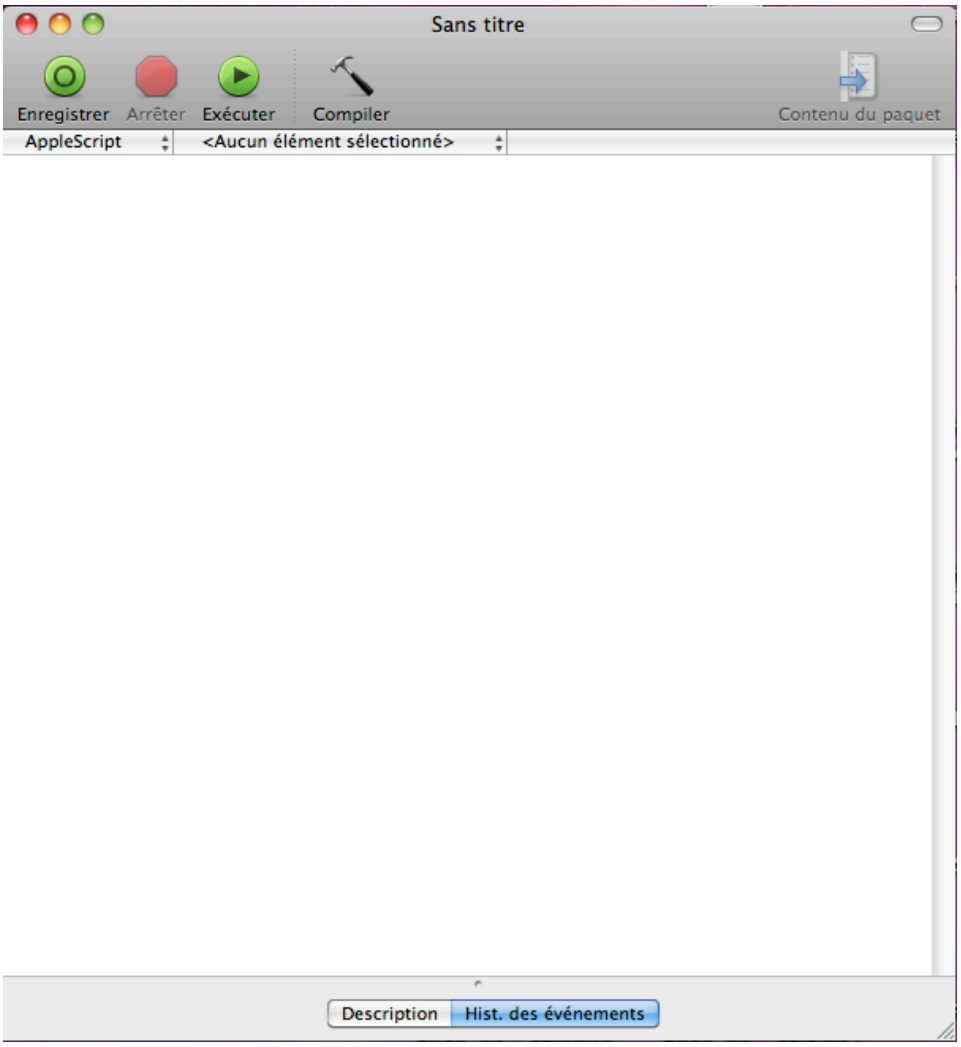
5. Automatisation de l'exportation de données vers Excel ou Numbers

La partie automatisation ne peut pas être géré directement par Real Basic, a plateforme Mac OS X ne prenant pas en charge l'automation OLE comme la plateforme Windows. Toutefois tout n'est pas perdu il suffit d'utiliser un des outils mis à notre disposition par Apple pour créer des Scripts : AppleScript.
Nous allons voir dans cette partie comment créer des scripts AppleScript et les intégrer au sein de Real Basic afin que celui-ci les reconnaissent comme des méthodes à part entière…

a. L'éditeur AppleScript

Vous trouverez cet éditeur sous SnowLeopard sous cet emplacement : Macintosh HD/Applications/Utilitaires/AppleScript. Pour ceux qui sont encore sous Leopard vous le trouverez sous Mac HD/Application/AppleScript.

Une fois l'éditeur lancé vous devriez voir apparaître la fanêtre suivante :



Vous êtes prêt à créer vos premiers scripts. Nous allons créer plusieurs petits scripts, qui vont nous permettre de créer des fonctions dans Real Basic pour des actions bien spécifiques :

- ***OpenExcel*** qui va permettre d'ouvrir l'application Excel et ouvrir une feuille de calcul que l'on activera.
- ***InsertHeader*** qui créera une entête pour chaque colonne créée.
- ***Fillcell*** qui va permettre de remplir chaque cellule du tableau avec une valeur générée aléatoirement.
- ***DoOperation*** qui va permettre d'effectuer une addition pour chaque colonne.
- ***QuitExcel*** qui va nous permettre de fermer l'application.

Tous ces scripts vont être créés deux fois une pourExcel et une fois pour Numbers.

b. Les commandes nécessaires à l'écriture de scripts

Les commandes de scripts utiles pour que chaque script puisse être utilisable avec Real Basic sont ***On Run*** et ***End Run***.
Pour appeler une application il faut utiliser la commande "***Tell application***" suivie du nom de l'application à ouvrir dans notre cas "***Microsoft Excel***". "***End Tell***" termine le bloc d'appel.
Pour activer l'application après l'avoir ouverte nous avons à notre disposition la commande "***Activate***".
Pour active rune application déjà ouverte il suffit de'utiliser la commande **activate** *application* "Microsoft Excel"
Pour ouvrir une feuille de calcul Excel par eemple "Feuil1" il suffit d'utiliser la commande "***Open***".
Pour resélectionner une feuille de calcul déjà ouverte il suffit d'utiliser la commande **activate object** *worksheet* "Feuil1".
On peut de même passer des paramètres à notre script ce qui nous permettra de créer une méthode paramétrée au sein de RealBasic. Il suffit pour ce faire de placer les paramètres après la commande On Run et ce entre crochets comme ceci : on run {i, j, valeur} par exemple.
Il est possible de faire référence à ses paramètres au sein du script comme dans une méthode pour remplir chacune des cellules d'un tableau Excel par exemple, comme ceci :

```
set value of cell i of column j of active sheet to valeur
```

Voilà toutes les commandes nécessaires à l'élaboration de nos scripts. Je vais donc vous lister les différents scripts qui vont être utilisés dans la suite de ce tutoriel.

c. Les scripts permettant d'ouvrir Microsoft Excel et Apple Numbers

Voici le script permettant d'ouvrir une instance de l'application Excel et d'ouvrir une feuille de calcul nommée "Feuil1 par défaut.

```
on run
	tell application "Microsoft Excel"
		activate
		open "Feuil1"
	end tell
end run
```

Voici maintenant le script permettant d'effectuer la même opération mais cette fois ci avec Numbers :

```
on run
	tell application "Numbers"
		activate
		tell document 1
			tell sheet 1
			end tell
		end tell

	end tell
end run
```

Comme vous pouvez le constater le script diffère quelque peu de celui associé à Excel. Il faut d'bord associer un nouveau document et après appeler une nouvelle feuille de travail.

d. Le script permettant de créer une entête de colonne

Ce script permet de créer une entête de colonne que l'on nommera Col suivi du numéro de colonne. Ce nombre est généré en fonction du paramètre renter par l'utilisateur.

```
on run {j, valeur}
	tell application "Microsoft Excel"

		activate application "Microsoft Excel"
		activate object worksheet "Feuil1"

		set value of cell 1 of column j of active sheet to valeur
	end tell
end run
```

La meme chose pour Numbers :

```
on run {j, valeur}
	tell application "Numbers"
		activate
		tell document 1
			tell sheet 1
				tell table 1
					set value of cell 1 of column j to valeur
				end tell
			end tell
		end tell
	end tell
end run
```

Comme vous pouvez le constater la plupart des operations s'effectuent avec les commandes tell pour Numbers, mis à part le remplissage d'une cellule qui ressemble énormément à la syntaxe utilisée pour Excel.

e. Le script permettant de remplir chaque cellule d'une feuille de calculs.

Nous allons maintenant introduire le script le plus important c'est à dire celui qui va nous permettre de d'introduire les données dans chaque cellule du tableau. Voici le script utilisé.

```
on run {i, j, valeur}
	tell application "Microsoft Excel"

		activate application "Microsoft Excel"
		activate object worksheet "Feuil1"

		set value of cell i of column j of active sheet to valeur
	end tell
end run
```

On retrouve bien la commande de script ***Set Value*** qui va se charger de compléter la ligne en cours de la feuille de calculs Excel.

Voici la version pour Numbers (voir page suivante) :

```
on run {i, j, valeur}
	tell application "Numbers"
		activate
		tell document 1
			tell sheet 1
				tell table 1
					set value of cell i of column j to valeur
				end tell
			end tell
		end tell
	end tell
end run
```

La syntaxe du set value est plus simple que pour Excel. Mais le script est un peu plus long il faut appeler chaque partie du document :

Document -> Feuille de calcul -> tableau -> Accès à la cellule

f. Le script permettant d'effectuer une operation sur une colonne

Nous allons maintenant écrire un script nous permettant d'effectuer soit une addition soit une soustraction ou encore une multiplication sur une colonne.
Voici le script pour Excel :

```
set k to 0
	tell application "Microsoft Excel"

		activate application "Microsoft Excel"
		activate object worksheet "Feuil1"
		repeat with k from ligneDebut to ligneFin - 1

			if TypeOperation is equal to "Addition" then
				set totalColonne to totalColonne + (value of cell k of
					column colonne)

			end if
			if TypeOperation is equal to "Soustraction" then
				if k is equal to ligneDebut then
					set totalColonne to value of cell k of column
						colonne
				else
					set totalColonne to totalColonne - (value of cell k
						of column)
```

```
                end if

            end if
            if TypeOperation is equal to "Multiplication" then
                if k is equal to ligneDebut then
                    set totalColonne to value of cell k of column
                                colonne of activesheet
                else
                    set totalColonne to totalColonne * (value of cell k
                                of column colonne)
                end if

            end if
        end repeat
        set value of cell ligneFin of column colonne of active sheet to
                totalColonne
    end tell
end run
```

Ce script fait montre comment declarer une variable dans un script. :

Set totalColonne to 0

Ce script fait également apparaître une boucle permettant de prendre chaque valeur de chaque colonne afin d'effectuer l'opération désirée.

Ensuite on inscrit le contenu de la variable totalColonne à la dernière ligne de la colonne.

Maintenant voici le scrit effectuant la meme operation mais cette fois ci pour Numbers.

```
on run {ligneDebut, ligneFin, colonne, TypeOperation}
    set totalColonne to 0
    set k to 0

    tell application "Numbers"
        activate
        tell document 1
            tell sheet 1
                tell table 1

                    repeat with k from ligneDebut to ligneFin - 1

                        if TypeOperation is equal to "Addition" then
                            set totalColonne to totalColonne + (value
                                    of cell k of column
                                    colonne)
```

```
                    end if
                    if TypeOperation is equal to "Soustraction"
                                then
                        if k is equal to ligneDebut then
                            set totalColonne to value of cell k of
                                                column colonne
                        else
                            set totalColonne to totalColonne -
                                                (value of cell k
                                                of column
                                                colonne)

                        end if

                    end if
                    if TypeOperation is equal to "Multiplication"
                                then
                        if k is equal to ligneDebut then
                            set totalColonne to value of cell k of
                                                column colonne
                        else
                            set totalColonne to totalColonne *
                                                (value of cell k
                                                of column
                                                colonne)
                        end if

                    end if
                end repeat

            end tell
            tell table 1


                set value of cell ligneFin of column colonne to
                                totalColonne
            end tell
        end tell
    end tell
end tell
end run
```

On retrouve les mêm differences que pour les precedents scripts, mais le principe reste le meme que la version Excel.

g. Le script de fermeture de l'application Excel

Voici le script demandant la fermeture de l'application Excel :

```
tell application "Microsoft Excel"
	quit
end tell
```

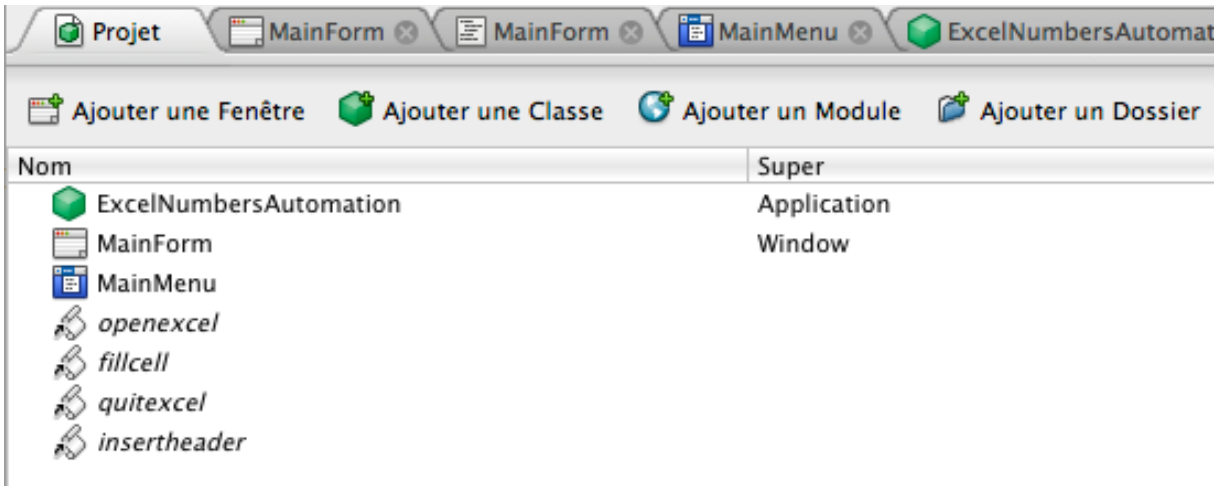
Le script de fermeture de l'application Numbers ne diffère que par le nom de l'application :

```
tell application "Numbers"
	quit
end tell
```

Comme vous pouvez le constater il suffit d'appeler la commande quit pour demander la fermeture d'une application.

h. *Intégration des scripts dans RealBasic 2009*

Pour intégrer ces scripts au sein de votre projet RealBasic il suffit d'effectuer un drag'n'drop de chaque fichier script vers la page de configuration de votre projet. Vous verrez ensuite apparaître dans la fenêtre de projet.



A noter que les scripts doivent être sauvegardés dans le même réperoire que votre projet, c'est-à-dire username/Documents/RealBasic.
Vos scripts maintenant peuvent être utilisés comme des méthodes dans RealBasic.

6. Code de l'application

a. *Introduction*

Nous allons maintenant finaliser le projet en introduisant le code nécessaire au bon déroulement de l'application. Il va falloir éditer le code pour le click de souris sur le

bouton Exporter et lorsque l'on clique sur la commande menu Export … Vers Excel ou Vers Numbers. Et pour finir nous verrons comment coder la méthode associer au code de fermeture de l'application. Il va nous falloir aussi écrire le code correspondant à la saisie du nombre de colonnes et du nombre de lignes. Il va falloir aussi éditer le code correspondant au type d'opération que nous allons effectuer sur une colonne.

b. *Initialisation des propriétés et des zones de texte*

Cette operation est réalisée dans la fonction Open() dans la section Events Handlers. Vous pouvez double cliquez sur la fiche dnas la page de conception pour activer l'éditeur de code sur cet événement.

Voici ce que nous allons initialiser :

```
txtColonnes.Text = "1"
txtNbLignes.Text = "1"
NbColonnes = 1
NbLignes =1
```

c. *Code correspondant à la saisie du nombre de colonnes et du nombre de lignes*

Voici le code qui va nous permettre de staocker le nombre de lignes dans la propriété NbLignes :

```
if IsNumeric(txtnbLignes.text)  Then
        NbLignes = CType(Val(txtnbLignes.Text), Integer)
Else

        MsgBox "Veuillez saisir un nombre SVP…"


End If
```

Voici maintenant le code qui va nous permettre de stocker le nombre de colonnes dans la propriété NbColonnes :

```
If IsNumeric(txtColonnes.text)  Then
        NbColonnes = CType(Val(txtColonnes.Text), Integer)
Else

        MsgBox "Veuillez saisir un nombre SVP…"


End If
```

Pour éditer ces deux échantillons de code il vous suffit d'éditer la zone de texte correspondant aà l'événement TextChange de vos composants zone d'édition de texte. La fonction ***CType*** permet de convertir une chaîne de caractères vers un entier dans notre cas. Nous faisons tout d'abord un test pour savoir si chaque caractère saisi est bien un caractère numérique par l'intermédiaire de la fonction IsNumeric.

d. Obtention du type d'opération à réaliser

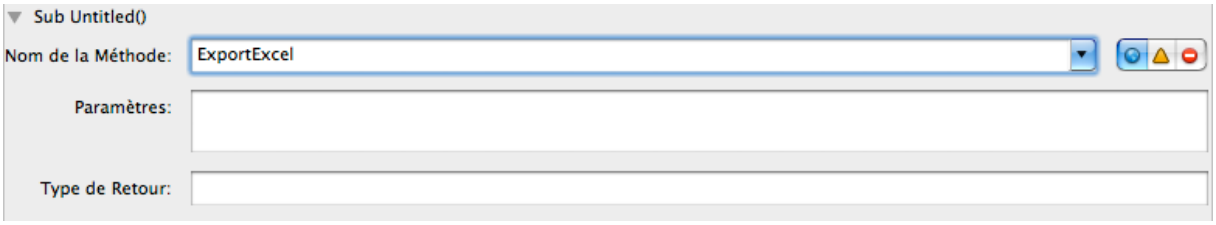
Nous allons pour celà éditer l'éditeur de code pour l'événement Change de notre composant cbOperation.

```
TypeOperation = cbOperation.Text
```

Nous initialisons donc la propriété TypeOperation avec la selection effectuée par l'intermédaiire de la combobox.

e. Exportation des données vers Excel

Nous allons créer une méthode pour pouvoir gérer l'exportation des données vers Excel, pour ce faire dans la page d'édition du code cliquez sur la commande Ajouter une Méthode. Un assistant s'ouvre et vous permet de personnaliser votre méthode :



Appelez la ExportExcel par exemple.
Nous allons y intrioduire le code suivant :

```
if (NbLignes <> 0) and (NbColonnes <> 0) then
        Dim i As Integer
        Dim j As Integer

        openexcel()


        for i = 0 to NbColonnes - 1
                insertheader(i+1, "Col" + str(i))
                for j = 0 to NbLignes - 1

                        Values = rnd*10000
                        fillcell(j+2, i+1, Values)
                        if  j = NbLignes – 1 then
```

```
                DoOperationExcel(2, NbLignes+2, i+1, TypeOperation)
                end if
            Next
        Next

        quitexcel()


    End if
```

Faites de meme pour Numbers. Appelez la fonction ExportNumbers :

```
        if (NbLignes <> 0) and (NbColonnes <> 0) then
        Dim i As Integer
        Dim j As Integer

        openNumbers()


        for i = 0 to NbColonnes - 1
            insertheaderNumbers(i+1, "Col" + str(i))
            for j = 0 to NbLignes - 1

                Values = rnd*10000
                fillcellNumbers(j+2, i+1, Values)
                if  j = NbLignes – 1 then

                DoOperationNumbers(2, NbLignes+2, i+1, TypeOperation)
                end if
            Next
        Next

        quitNumbers()


    End if
```

Comme vous pouvez le constater nous pouvons avoir accès à nos fonctions écrites avec l'éditeur de scripts AppleScript comme n'importe quelle méthode créé sous RealBasic. Nous retrouvons bien les quatre fonctions defines sous forme de scripts dans cette fonction. Les valeurs entières sont créés aléatoirement grace à la méthode rnd.
La méthode quitExcel oblige l'utilisateur a confirmé l'enregistrement de la feuille de calcul (ou non) avant de fermer l'application Excel. Si vous ne voulez pas de ce comportement, il vous suffit de ne pas appeler cette méthode, comme celà la feuille Excel restera ouverte.

Maintenant que la méthode est créée il va falloir la rattacher à la commande de menu Export… Vers Excel ainsi qu'au clic de souris sur le bouton ExportExcel.

Pour ce faire il faut éditer le code correspondant à cet événement il vous suffit de double-cliquer sur le bouton Exporter.
Il vous suffit ensuite d' appler la méthode ExportExcel comme ceci :

ExportExcel()

Il faut ensuite faire de même avec la commande de menu. Il suffit de double créer un Gestionnaire pour le menu. Pour ce faire il suffit d'appuyer sur le bouton de commande "**Ajouter un Gestionnaire pour le menu**" dans la fenêtre d'édition du code. Dns l'assistant qui s'affiche il suffit de sélectionner l'item de menu correspondant à l'handler que vous souhaitez créer (c'est-à-dire MenuVersExcel). Dans la partie code il vous suffit d'appeler la méthode que vous venez de créer.
La meme operation est à réaliser pour Numbers.

Ce tutoriel est à present terminé, j'espère que vous en saurait plus sur l'interaction entre AppleScript et RealBasic permettant d'automatisr des applications bureautiques de manière très simple ma foi. De plus ces scripts sont réutilisables pour vos prochains travaux. Pour pouvoir l'exécuter de manière autonome il faut générer l'application par l'intermédiaire de la commande se situant dans la barre d'outils principale de RealBasic. Il crée alors un sous repertoire Builds dans lequel il a créé votre exécutable .app.

IV CONCLUSION

Ce petit tutoriel vous a montré comment utiliser AppleScript afin de créer des méthodes que RealBasic. Cetutoriel vous a montré de meme qu'il est facile d'automatiser des applications bureautiques telles que Microsoft Excel et Apple Numbers. L'exemple pris est très simple, c'est-à-dire prendre une série de chiffres aléatoirement, mais vous n'aurez aucun mal à l'adapter à d'autre source de données.