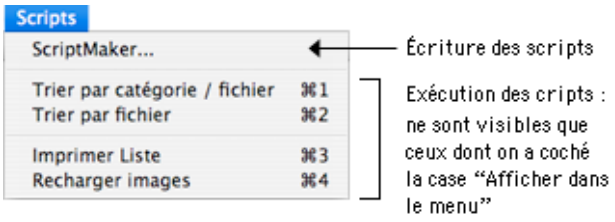
1. Les scripts en FileMaker : principe

Les scripts sont très utiles pour automatiser les tâches complexes ou celles que l'on doit répéter très souvent.

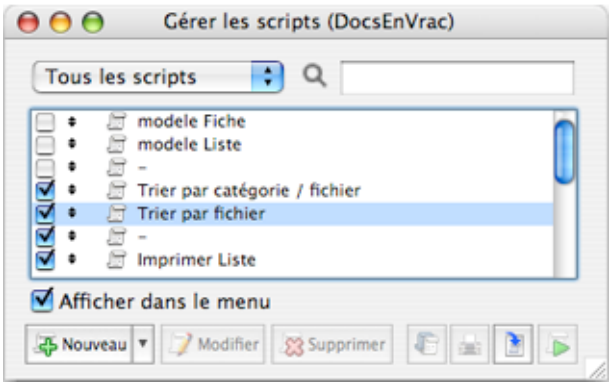
Un script contient la succession des commandes à exécuter, pas à pas, pour accomplir la tâche voulue : quand l'utilisateur demande l'éxécution du script, les instructions qu'il renferme sont éxécutées une à une, dans l'ordre où elles ont été placées.



La commande scriptMaker du menu Script

C'est elle qui permet de définir des scripts. Elle ouvre un dialogue nommé "Gérer les styles" (photo ci-contre).

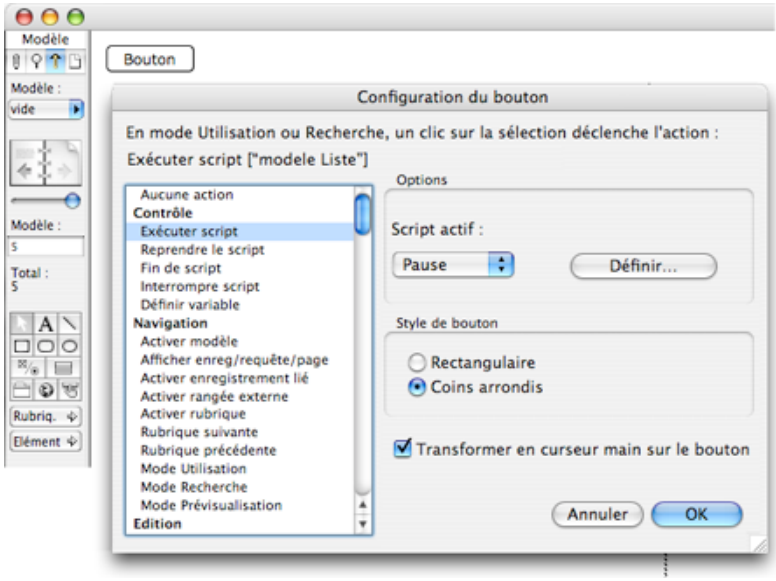
Noter la boîte à cocher, à gauche de chaque script : elle détermine si le script apparaît dans le menu Scripts ou s'il est masqué.



Lancement d'un script

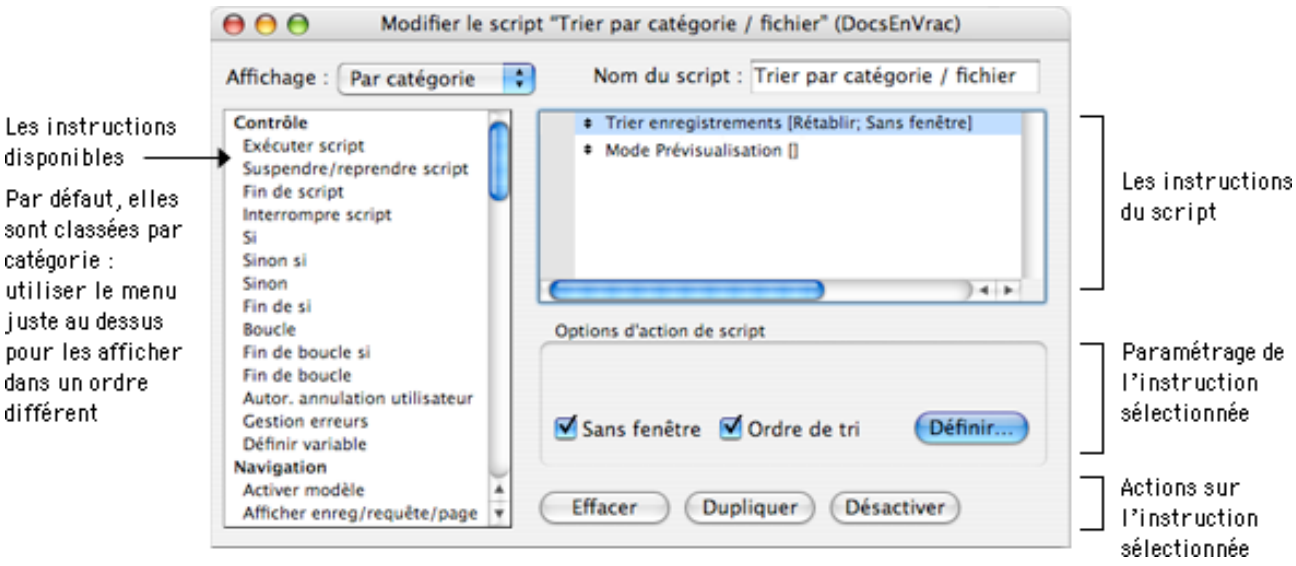
Un script peut être lancé directement en le sélectionnant dans le menu Scripts (à condition qu'il ne soit masqué, bien entendu).

Mais on peut également poser un bouton sur un modèle et lui associer le script : en mode utilisation, l'utilisateur n'aura qu'à cliquer sur le bouton pour déclencher le script.



Le dialogue de définition d'un script

On peut utiliser dans les scripts toutes les commandes de FileMaker, mais également des instructions supplémentaires qui ne sont pas disponibles dans les menus du logiciel. *Un script peut lui-même faire appel à un un sous-script, grâce à l'instruction "Exécuter script".*



2. Mise au point de scripts simples (quelques pistes…)

Actions de script utiles pour des scripts basiques

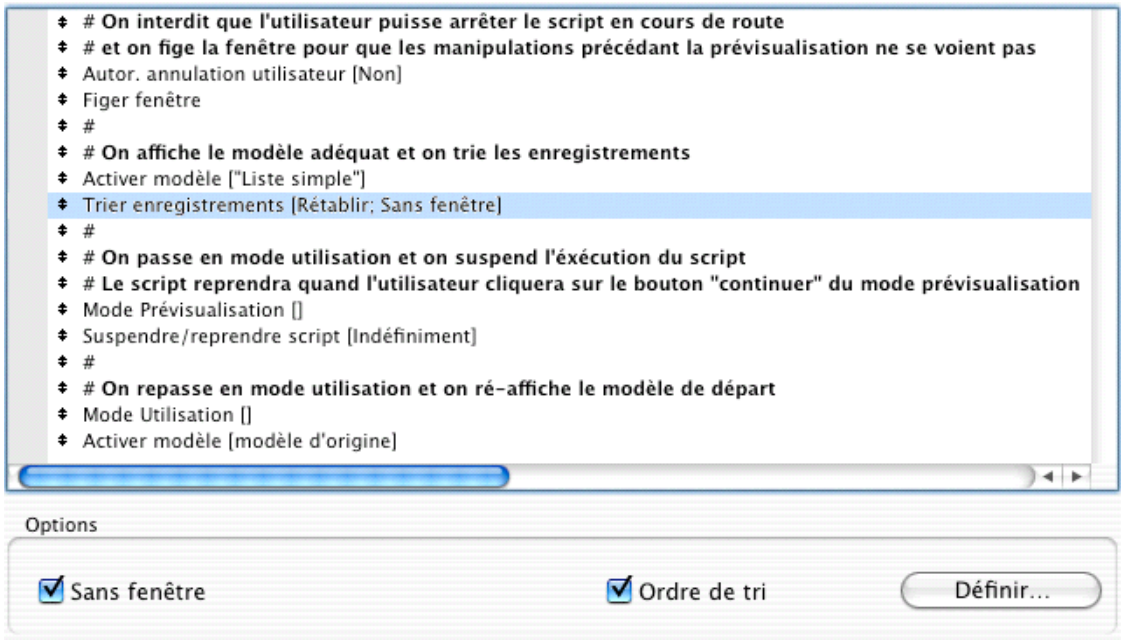
Pour :	*Catégorie d'instructon*	*Action de script*
Changer de mode Changer de modèle	*Navigation*	Mode utilisation / prévisu / recherche / modèle Activer modèle
Afficher tous les enregistrements Inverser enregistrements trouvés/ignorés	*Enreg. trouvés*	Afficher tous les enregistrements Afficher enregistrements ignorés
Trier selon un ordre prédéfini Remettre dans l'ordre originel	*Enreg. trouvés*	Trier enregistrements Annuler tri des enregistrements
Importer / exporter les données	*Enreg.*	Importer enregistrements Exporter enregistrements
Appeller un sous-script dans un script	*Contrôle*	Exécuter script
Insérer un commentaire	*Divers*	Commentaire

(Voir aussi les actions de script de la catégorie Fenêtres)

Un peu plus complexe :
exemple de script pour faciliter l'impression des données

Le scénario que l'on veut obtenir est le suivant :

- Préparation du document : choix du modèle et tri des enregistrements
- Prévisualisation et arrêt momentané du script : on rend la main à l'utilisateur pour qu'il puisse observer le document et l'imprimer s'il le désire (la zone d'état affichera alors les boutons Continuer et Annuler).
- Poursuite de l'exécution quand l'utilisateur clique sur Continuer : on remet la fenêtre dans la situation de départ (retour au mode utilisation et au modèle affiché précédemment).



Remarque :

L'action "Suspendre/reprendre le script [indéfiniment]", fait que le script s'arrête et attend que l'utilisateur décide de la suite des opérations. Normalement, l'utilisateur dispose alors de deux boutons dans la zone d'état : "Continuer" (pour lancer la poursuite du script) et "Annuler" (pour interrompre le script sans exécuter les instructions restantes).

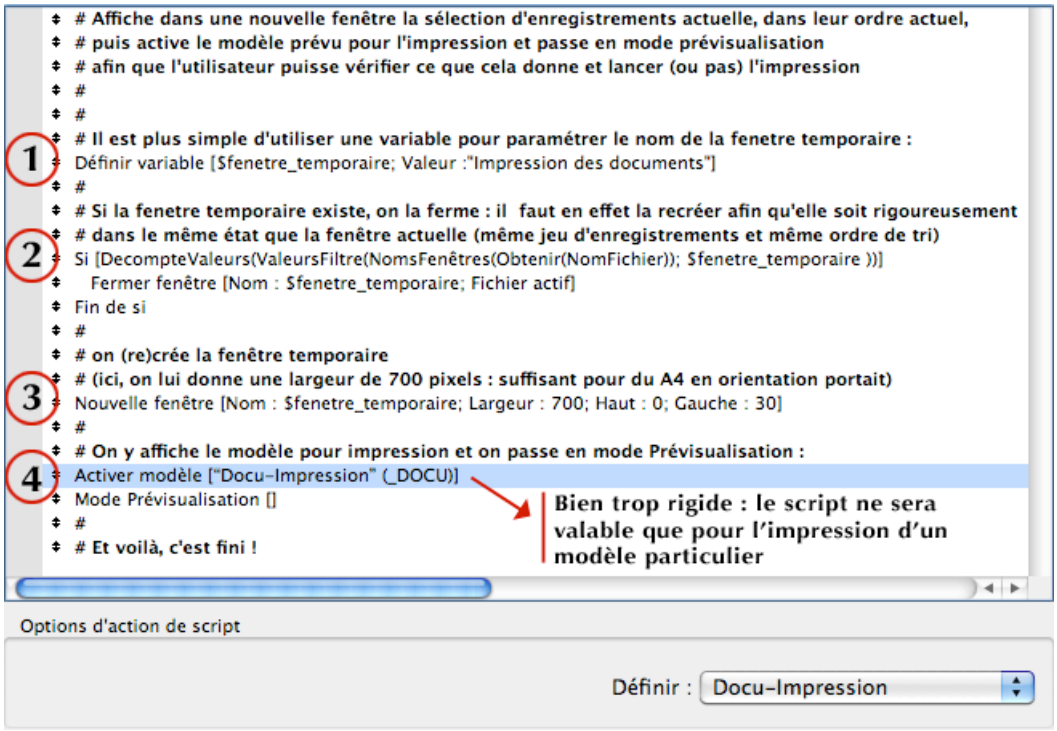
Mais ici, on veut éviter que l'utilisateur interrompe le script avant que l'on soit revenu en mode utilisation et sur le bon modèle. Pour ce faire, on utilise l'instruction "Autoriser annulation utilisateur [Non]" en début de script : cette instruction fait que Filemaker n'affichera pas le bouton "Annuler" dans la zone d'état : l'utilisateur ne disposera que du bouton "Continuer" qui provoque la reprise du script. De cette manière, on est sûr que les instructions suivant la suspension du script seront exécutées.

3. Scripts un peu plus rusés (variables, instructions de contrôles, paramètres)

Reprenons un exemple de script d'impression, légèrement différent. Ce que l'on veut que le script fasse cette fois :

- Ouvrir une nouvelle fenêtre pour que l'utilisateur prévisualise le document
- Activer le modèle d'impression, passer simplement en mode prévisualisation et arrêter (puisque la prévisu se fait dans une nouvelle fenêtre, l'utilisateur n'aura qu'à la fermer quand il aura fini).

Exemple 1 – Variables, structures de contrôle



1. On utilise ici une variable ($fenetre_temporaire) pour stocker le nom de la nouvelle fenêtre, pour simplifier l'écriture des instructions qui utilisent ce nom dans la suite du script d'une part (*cf.* 2, 3), mais surtout *pour pouvoir en changer facilement par la suite* si besoin.

   Une variable est comme une "case" dans la mémoire de l'ordinateur où l'on peut écrire une valeur temporairement. Celle que nous utilisons ici est une variable locale, c'est-à-dire qu'elle n'existe que le temps d'exécution du script. En Filemaker, le nom de ces variables commence par un $.

2. À cette étape, on veut agir différemment selon que la fenêtre à ouvrir existe déjà pas : si elle existe, comme on ne maîtrise pas ce qui y est affiché, il faut la fermer et la re-créer afin que la nouyvelle fenêtre soit dans la même situation que la fenêtre depuis laquelle on a demandé l'impression. Pour ce faire, on utilise l'*instruction de de contrôle* :

   SI *test* **ALORS** *ce qu'il faut faire si le test est vérifié* **SINON** *ce qu'il faut faire s'il ne l'est pas* **FIN DE SI**

   sans sa composante facultative "Sinon…", inutile ici. Selon la situation, le résultat du test sera faux (0) ou vrai (1), et l'instruction de femeture de la fenêtre ne sera exécutée que dans ce dernier cas. La formule pour tester l'existence d'une fenêtre est : **DecompteValeurs(ValeursFiltre(NomsFenêtres(Obtenir(NomFichier));** $fenetre_temporaire**))**, où $fenetre_temporaire (la variable créée juste avant) contient le nom de fenêtre que nous voulons tester.

3. Il ne reste plus qu'à créer la nouvelle fenêtre en lui donnant pour nom la valeur mémorisée par $fenetre_temporaire…

4. … et à passer en mode prévisaulisation après avoir activé le modèle adéquat. Et là, nous avons un problème : notre script tourne bien, mais il n'est valable que pour imprimer un modèle particulier ce qui n'est vraiment pas satisfaisant : nous verrons page suivante comment lui injecter un peu plus de souplesse.

Nouvelles instructions vues dans ce script :

Pour :	***Catégorie d'instruction***	
Définir une variable	*Contrôle*	
Si … Alors … Sinon	*Contrôle*	(Il suffit de choisir l'instruction SI pour mettre d'un coup SI et FIN DE SI)
Fermer fenêtre	*Fenêtres*	
Nouvelle fenêtre	*Fenêtres*	

Exemple 2 – Comment passer un paramètre au script pour gagner en souplesse

Nous voulons maintenant faire en sorte que ce script soit valable quel que soit le modèle à imprimer. La solution :

- Appeler le script en lui passant un paramètre pour lui indiquer quelle impression on veut obtenir. Pour faire simple, décidons que ce paramètre sera un "code" sous forme de chaîne de caractère.
- Ajouter au script : - une instruction qui 'capte" ce paramètre au début du script,
  - une instruction de contrôle pour activer le bon modèle en fonction de l'impression demandée.

```
# On récupère dans une variable la valeur passée en paramètre,
# via la fonction Obtenir( ParamètreScript )
1 Définir variable [$imprimer_quoi; Valeur :Obtenir( ParamètreScript )]
#
# Du coup, le nom de la fenetre temporaire peut varier selon l'impression demandée :
2 Définir variable [$fenetre_temporaire; Valeur :Cas(  $imprimer_quoi = "auteurs" ; "Impression des auteurs"; $im
#
# Si la fenetre temporaire existe, on la ferme puis on la (re-)crée
Si [DecompteValeurs(ValeursFiltre(NomsFenêtres(Obtenir(NomFichier)); $fenetre_temporaire ))]
   Fermer fenêtre [Nom : $fenetre_temporaire; Fichier actif]
Fin de si
Nouvelle fenêtre [Nom : $fenetre_temporaire; Largeur : 700; Haut : 0; Gauche : 30]
#
# On y affiche le modèle voulu par test sur $imprimer_quoi et on passe en mode Prévisualisation :
3 Si [$imprimer_quoi = "auteurs"]
   Activer modèle ["Pers-Liste" (_PERS)]
Sinon si [$imprimer_quoi = "docu-Listing"]
   Activer modèle ["Docu-ImprListe" (_DOCU)]
Sinon si [$imprimer_quoi = "docu-Detail"]
   Activer modèle ["Docu-ImprFiche" (_DOCU)]
Sinon
   Activer modèle ["FILE fichiers sur disque" (_FILE)]
Fin de si
# et on passe en mode Prévisualisation
Mode Prévisualisation []
```

1. On met dans la variable $imprimer_quoi le code passé en paramètre, qu'on obtient par à un calcul très simple : il suffit d'utiliser la fonction[1] : Obtenir(ParamètreScript).

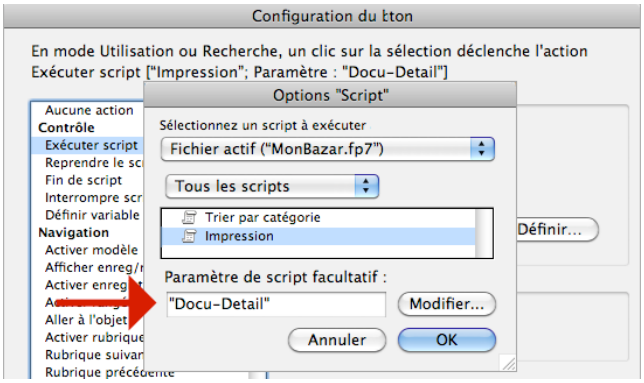
2. Tant qu'à faire, on peut aussi donner à la fenêtre un nom différent selon l'impression demandée, si on le souhaite : il suffit pour cela d'utiliser de calculer ce nom à l'aide de la fonction :

CAS(	$fenetre_temporaire = Cas(
test1 ; valeur si test1 est vrai ;	*$imprimer_quoi = "auteurs"*; "Impression : auteurs" ;
test2 ; valeur si test2 est vrai ;	*$imprimer_quoi = "docu-Listing"*; "Impression – Liste documents" ;
test3 ; valeur si test3 est vrai ;	*$imprimer_quoi = "docu-Detail";* "Impression – Fiches docs" ;
valeur si aucun test n'est vérifié	"Impression – Liste des fichiers"
)	)

3. Après avoir ouvert la nouvelle fenêtre, on choisit le modèle à activer grâce à une variante de l'instruction SI… ALORS… SINON… qui permet d'enchaîner une cascade de tests :

SI *$imprimer_quoi = valeur 1*
  Activer *modèle 1*
SINON SI SI *$imprimer_quoi = valeur 2*
  Activer *modèle 2*
SINON SI SI *$imprimer_quoi = valeur 3*
  Activer *modèle 3*
FIN de SI

Pour finir, il n'y a plus qu'à ajouter, dans le réglage de chaque bouton qui déclenchent le script, le paramètre à lui passer :



Et le tour est joué !

[1] Dans le dialgue d'écriture des formules, il faut aller chercher cette fonction dans la catégorie "Obtention". On peut aussi passer plusieurs paramètres à un script, mais dans ce cas la formule permettant de les récupérer sera plus complexe.