NOTE DE PROGRAMMATION

Vous trouverez dans ce document toutes explications concernant la programmation de l'outil de gestion. Cette note se divise en deux parties ; la première pour la création de l'univers d'investissement, et la seconde pour la création de l'outil de gestion.

I. UNIVERS D'INVESTISSEMENT

L'univers d'investissement est composé de cinq indices boursiers (issus de l'archive de données fournie) et de dix obligations. Il est construit par le code VBA du fichier *'FinalData.xlsm'*, plus précisément par la procédure main.

Premièrement, la procédure *'creation_fichier_data'* va créer le fichier résultat et reporter les dates et les cours des indices sur celui-ci. Pour faire ceci, on demande à l'utilisateur de sélectionner tour à tour les fichiers de données des indices. Pour chacun de ses fichiers, le code fait appel à la fonction *'fn_recup_data'* qui attribue la plage de données à une variable. Une fois les plages récupérées, on les reporte dans le fichier résultat.

Deuxièmement, la procédure *'data_oblig'* va ouvrir le fichier résultat pour continuer de le construire en y reportant les obligations générées dans cette procédure. Le module de classe *'ClasseOblig'* est utilisé afin de définir l'objet *'Ob'*, qui correspond à une obligation, ainsi que toutes ses caractéristiques. Dans une boucle sur le nombre d'obligations à créer, le code répète trois étapes. D'abord un objet *'Ob'* est créé et ajouté à la collection. Ensuite, les caractéristiques de l'obligation sont définies aléatoirement ou selon les contraintes imposées par le client. Pour finir, les caractéristiques de l'obligation sont reportées sur la seconde feuille du fichier résultat. Le fichier résultat 'finaldata.xls' est désormais terminé.

Enfin, la procédure *'xlToCsv'* va ouvrir le fichier résultat tout juste terminé et va transformer chacune de ses deux feuilles en un fichier CSV. L'exécution de cette procédure est dédiée à construire une base de données MySQL à partir de l'univers d'investissement. Après importation des CSV dans MySQL, on pourra ainsi faire des requêtes sur les données de l'univers d'investissement.

II. OUTIL DE GESTION

L'outil de gestion donne les informations générales du portefeuille du client, ainsi que ses statistiques à la date de la requête en cours, à la fois depuis la création du portefeuille et depuis la dernière requête effectuée. Il est construit par le code VBA du fichier *'OutilGestion.xlsm'*, plus précisément par la procédure main.

Tout d'abord, on demande à l'utilisateur de rentrer la date de la requête actuelle et de la dernière requête s'il y en a une, et on les stocke dans des variables. On utilise ces dates tout au long du code. Pour les statistiques depuis la création du portefeuille, évidemment, seule la date de la requête actuelle sera utilisée. En revanche pour les statistiques depuis la dernière requête, deux méthodes sont utilisées à différents endroits du code. Si possible, on utilise les deux dates pour faire un calcul directement sur la période entre les deux. Lorsque ce n'est pas approprié ou moins instinctif, le code calcule sur la période de la création du portefeuille jusqu'à la dernière requête, puis calcule la différence avec la statistique à la date de la requête actuelle.

Juste après ça, on détermine la composition de base du portefeuille, c'est-à-dire le montant initial investi par le client, et on pose un arbitrage entre les deux poches.

On s'occupe ensuite des indices, dont la plages de cours est récupérée, en fonction des dates rentrées par l'utilisateur. Pour chaque colonne de la plage, soit pour chaque titre, on calcule les indicateurs de performance et de risque, notamment avec l'aide des fonctions *'fn_variant_to_double'* qui change le type de chaque colonne de la plage en type double, et *'fn_rendements'* qui calcule le vecteur des rendements pour chaque vecteur de cours. On reporte finalement chaque indicateur calculé sur la feuille de résultat.

La prochaine partie se concentre sur les obligations. On récupère les statistiques des obligations sur la seconde feuille du fichier de données, et on les stocke dans les paramètres d'un objet *'Ob'*, encore une fois par le biais d'un module de classe *'ClasseOblig'*. Cependant, comme les obligations ont été générées aléatoirement, elles ne finissent pas obligatoirement toutes dans le portefeuille. Pour les sélectionner, après les avoir rangées par taux d'intérêt décroissant, on fait une boucle sur ces obligations et on les choisit jusqu'à ce que la somme des montants nominaux dépasse le montant d'investissement de la poche obligations. Par ailleurs, toute la somme prévue pour les obligations n'est pas forcément utilisée, la différence est donc placée dans une poche cash en euros.

Pour chaque obligation sélectionnée, après avoir attribuer à chacun de ses paramètres la valeur correspondante provenant du fichier de données, on va calculer sa valeur à la date de la requête actuelle, et en parallèle à la date de la dernière requête par le même processus. On calcule le nombre de coupons déjà détachés pour chaque date avec l'aide de la fonction *'fn_ajouter_temps'*. À partir de ce nombre, on appelle la fonction *'valeur_act'* du module de classe pour calculé la valeur de l'obligation pour calculer sa valeur à cette date.

En parallèle, on détermine également si le remboursement du montant nominal a déjà été effectué ou non, en fonction de si la valeur de l'obligation est nulle ou non, et on calcule les intérêts reçus par les coupons et éventuellement le remboursement. On reporte pour finir les résultats sur la feuille résultat, et on crée une plage de résultat avec la synthèse des trois poches, résumant le portefeuille entier.

Enfin, on appelle la procédure de style *'mise_en_page'*, qui prend la feuille résultat en argument afin de la styliser. On prend aussi le nombre d'obligations sélectionnées pour que l'encadrement des cellules soit juste.