La gestion des données créatives

Extensis™

Portfolio 8™

Guide Visual Basic

Contact

Extensis

1800 SW First Avenue,
Suite 500
Portland, OR 97201 États-Unis
Toll Free: (800) 796-9798
Tél. : (503) 274-2020
Fax : (503) 274-0530
http://www.extensis.com

Extensis Europe

First Floor, Century House
The Lakes
Northampton NN4 7SJ
United Kingdom
Tél. : +44(0)1604 636 300
Fax +44 (0)1604 636 366
info@extensis.co.uk

Celartem, Inc.

Tél. : +81 3 5574 7236
Adresse Web/électronique : sales_ap@celartem.com
http://www.celartem.com/jp/

Service clientèle

Site Web/E-mail : *http://www.extensis.com/customerservice/*
Tél. : (800) 796-9798

Assistance technique

Site Web/E-mail : *http://www.extensis.com/support/*

Vente aux professionnels

Site Web : *http://www.extensis.com/corporatesales/*
Tél. : (800) 796-9798, ask for Corporate Sales

Commentaires sur la documentation

Site Web : *http://www.extensis.com/helpfeedback/*

Table des matières

Visual Basic et Portfolio

Ce document présente brièvement les commandes de Portfolio disponibles sur l'interface d'automatisation d'Exensis Portfolio 8.0 pour Windows. Les exemples dont il est question ici, et introduits en tant qu'exemples d'applications, ont été écrits sous Microsoft Visual Basic 6. Toutefois, les utilisateurs qui maîtrisent d'autres langages pouvant utiliser la fonction d'automatisation ne devraient pas avoir de difficulté à transcrire ces exemples dans le langage qu'ils utilisent. Ce document présume que les utilisateurs maîtrisent un tant soit peu Visual Basic.

Ce document ne constitue pas une introduction à Visual Basic ou un didacticiel ayant pour fonction d'expliquer comment scripter Portfolio pour exécuter certaines tâches. Pour en savoir plus sur les notions de base concernant Visual Basic, rendez-vous sur le site *http://msdn.microsoft.com/vbasic/*. Ce site contient de nombreuses informations utiles sur Visual Basic, ainsi que des liens vers d'autres sites Web traitant de la programmation. Pour en savoir plus sur le scriptage d'Extensis Portfolio, veuillez consulter les fichiers supplémentaires qui figurent sur le CD d'Extensis ou qui vous ont été fournis lorsque vous avez téléchargé votre produit.

Pour en savoir plus sur Portfolio, veuillez consulter le Guide de l'utilisateur Porfolio installé avec le client Porfolio. Pour obtenir les informations les plus récentes concernant Portfolio, rendez-vous sur le site Web d'Extensis à l'adresse suivante : *http://www.extensis.com/portfolio/*.

Introduction à l'interface d'automatisation de Porfolio

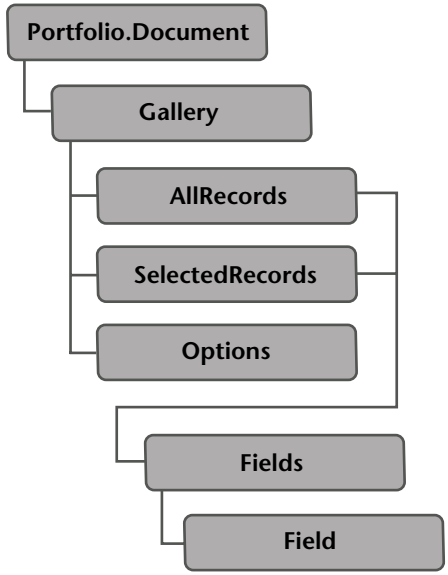
Ajout de Portfolio à votre projet Visual Basic

Tous les objets, propriétés et méthodes de l'interface de Portfolio (telle que décrite par la suite dans le présent document) sont pris en charge dans l'interface d'automatisation de Portfolio_V8.

Pour pouvoir utiliser l'objet Portfolio et sa famille de classes dans votre code Visual Basic, vous devez d'abord l'inclure dans les options de référence de votre projet VB. Utilisez la boîte de dialogue Référence des projets de l'interface de VB pour sélectionner la bibliothèque Portfolio_V8. Une fois cette opération réalisée, votre application VB reconnaîtra l'interface de document de Portfolio et vous pourrez référencer la famille de classes dans votre code VB.

Modèle objet de Portfolio :

Le diagramme ci-dessous montre la structure relative des classes du **Modèle objet Library de Portfolio**.



Vous pourrez remarquer qu'il n'existe aucun objet ou classe Catalog dans le modèle objet de Portfolio. L'objet document de Portfolio est en fait une collection de présentations. Si vous avez besoin de référencer le nom du catalogue, vous pouvez utiliser la propriété de nom `Gallery.Catalog`.

Création d'objets Portfolio

Comme pour tout accès à une application par l'interface d'automatisation, vous devez créer une variable pour stocker l'objet qui fait référence à la classe `Portfolio _ V8.document`, puis créer l'objet à l'aide de la commande définie

Par exemple :

```
Dim PortObj as Portfolio _ V8.document

Set PortObj = New Portfolio _ V8.Document
```

Si Portfolio est déjà activé, l'application VB utilisera cette instance de l'application. Portfolio est une application à instance unique, par conséquent il n'est pas possible de forcer une nouvelle instance de l'application à partir de VB.

Pour utiliser la classe Gallery, créez une variable d'objet pour stocker une référence à la classe `Portfolio _ V8.Gallery`, puis créez cette référence à l'aide de la commande définie.

Par exemple :

```
Dim PortGal as Portfolio _ V8.Gallery

Set PortGal = New Portfolio _ V8.Gallery
```

Dans la majeure partie de ce document, les exemples font référence à des objets nommés `PortObj` et `PortGal`. Il s'agit des objets décrits ci-dessus, et qui représentent les objets `Portfolio _ V8.Document` et `Portfolio _ V8.Gallery`. Ces objets sont les interfaces de communication qui permettent à votre code VB d'accéder aux fonctions et fonctionnalités de Portfolio.

Ouverture de catalogues

Pour ouvrir un catalogue Portfolio depuis l'interface d'automatisation, utilisez la fonction Open de l'objet Document. Si le catalogue a déjà été ouvert avec cet exemplaire de Portfolio (par script ou manuellement), la première présentation du catalogue apparaîtra en premier plan et deviendra la présentation active.

```
Function Open(Path As String, AccessMode
As Integer, Password As String) As
Integer
```

Les valeurs acceptables pour AccessMode sont les suivantes :

```
READER = 1

EDITOR = 2

PUBLISHER = 3

ADMINISTRATOR = 4
```

Par exemple :

```
x = PortObj.Open("C:\test.fdb",4,"")
```

Utilisation de catalogues

Ouverture de catalogues sur un serveur

Pour ouvrir un catalogue géré par un serveur Portfolio, utilisez la fonction OpenServer de l'objet Document. La commande accepte l'adresse du catalogue sous la forme « serveur/catalogue », où « serveur » est le nom du serveur ou son adresse IP.

```
Function OpenServer(Path As String,
AccessMode As Integer, Password As
String) As Integer
```

Les valeurs acceptables pour AccessMode sont les mêmes que celles indiquées ci-avant. Toutefois, n'oubliez pas que, en mode Administrateur, vous ne pouvez pas ouvrir les catalogues gérés par serveur avec votre code VB.

Par exemple :

```
x = PortObj.OpenServer( "192.0.0.0/Test.
fdb",3,"")
```

Ouverture de catalogues avec un utilisateur spécifié

Pour ouvrir un catalogue avec un utilisateur spécifié, utilisez la fonction Document.OpenByUserName pour les catalogues situés sur le disque, ou la fonction Document.OpenServerByUserName pour les catalogues servis (natifs ou par le service SQL). Le catalogue s'ouvre automatiquement en mode d'accès maximum tel que défini pour l'utilisateur spécifié.

```
Function OpenByUserName(Path As String,
UserName As String, Password As String)
As Integer

Function OpenServerByUserName(Path As
String, UserName As String, Password As
String) As Integer
```

Par exemple :

```
x = PortObj.OpenByUserName("C:\test.
fdb","User1", "password")

x = PortObj.OpenServerByUserName("192.0.0
.0/Test.fdb","User1","password")
```

Création de catalogues

Pour créer un nouveau catalogue, utilisez la nouvelle commande. Par défaut, le catalogue s'ouvre en mode Administrateur.

```
Function New(Path As String) As Integer
```

Par exemple :

```
x = PortObj.New("C:\test.fdb")
```

Utilisation de présentations

Lorsqu'un catalogue a été ouvert ou créé, la plupart des fonctionnalités du modèle de document Portfolio sont gérées par l'objet Gallery. Comme nous l'avons déjà mentionné, il n'existe aucun objet Catalog.

Sélection d'une présentation

La majeure partie des opérations effectuées dans votre code seront basées sur une référence à un objet `Gallery`. Cette référence renseigne Portfolio sur les enregistrements à manipuler pour le catalogue, mais aussi sur l'ensemble d'enregistrements visibles (images) à cibler. Pour définir quelle présentation est ciblée avec votre code, vous devez référencer un index avec l'objet Gallery :

```
PortObj.Gallery(x)
```

Pour connaître le nombre de présentations ouvertes dans le document Portfolio, utilisez la propriété `Count` de l'objet `Document` :

```
gCount = PortObj.Count
```

Pour déterminer la présentation active, utilisez la fonction `GetActive` de l'objet Document. Celle-ci vous donne le nom de la présentation active. Si aucune présentation n'est ouverte lorsque cette fonction est appelée, l'interface d'automatisation Visual Basic indiquera une erreur. Reportez-vous à l'Annexe D pour obtenir la liste complète des codes d'erreur.

```
sActive = PortObj.GetActive
```

Pour lire le nom d'une chaîne de la présentation, utilisez la fonction `GetGalleryIndexFromName` de l'objet Document.

```
Function GetGalleryIndexFromName(Gallery
Name As String) As Integer
```

Les exemples suivants se rapportent à la présentation active :

Par exemple :

```
PortObj.Gallery(PortObj.GetGalleryIndexFr
omName(PortObj.GetActive))
```

Tri d'une présentation

Vous pouvez utiliser la fonction tri (Sort) de l'objet `Gallery` pour organiser les enregistrements de la présentation active selon un ordre spécifique. Assurez-vous de n'utiliser pour le tri que des champs indexés et à valeur unique. Pour déterminer l'ordre de tri, définissez l'attribut `Direction` sur `True` pour trier selon un ordre ascendant, et sur `False` pour trier selon un ordre descendant.

```
Function Sort(FieldName As String,
Direction As Boolean) As Boolean
```

Par exemple :

```
x = PortObj.Gallery(1).
Sort("Filename",True)
```

Enregistrement d'une présentation

Pour enregistrer une présentation existante qui a été modifiée, utilisez la fonction `Gallery.Save`, ou utilisez la fonction `Gallery.SaveAs` pour enregistrer de nouvelles présentations.

```
Function Save() As Integer

Function SaveAs(sNewName As String) As
Integer
```

Par exemple :

```
x = PortGal.Save()

x = PortGal.SaveAs("gallery")
```

Ajout de fichiers à une présentation

Pour ajouter des fichiers source à une présentation, utilisez la fonction Catalogue de l'objet `Gallery`. La commande Catalogue observe tous les paramètres trouvés dans la boîte de dialogue **Options de catalogage** de l'application Portfolio.

```
Function Catalog(Path As String,
IncludeDirs As Boolean) As Boolean
```

Par exemple :

```
x = PortObj.Gallery(1).Catalog("C:\
Images\", True)
```

Vous pouvez utiliser n'importe quel chemin d'accès pour la chaîne ; ainsi, il est très facile de cataloguer des dossiers ou des volumes entiers, aussi bien que des fichiers individuels. Pour cataloguer les sous-dossiers d'un dossier ou d'un volume, définissez la propriété `IncludeDirs` sur `True`.

Catalogage - Options avancées

Accès aux options avancées

Vous pouvez également gérer les options de catalogage depuis votre application Visual Basic en utilisant l'interface d'automatisation de Portfolio. Pour y accéder, utilisez la classe AdvancedOptions de l'objet Portfolio. L'objet AdvancedOptions contient les propriétés qui correspondent aux commandes disponibles dans les boîtes de dialogue **Options de catalogage** et **Options avancées**. Vous les trouverez dans le menu Catalogue de l'interface de Portfolio.

Liste des propriétés du catalogue

Les propriétés disponibles pour la classe sont les suivantes :

```
AppendDescription
ExtractMetaData
ExtractThumbnail
SkipFiles
MergeKeywords
ParseKeywords
PathKeywords
   0 - None
   1 - File Name
   2 - File and Folder Name
   3 - Path Name
   4 - Path Name and Volume
ThumbnailSize
   0 - 112 x 112
   1 -256 x 256
```

L'exemple suivant ordonne à Portfolio de tenter d'extraire d'un fichier source une miniature incorporée, la prochaine fois qu'un fichier sera catalogué dans la présentation.

Par exemple :

```
PortObj.Gallery(1).AdvancedOptions.
ExtractThumbnail = True
```

Utilisation d'enregistrements et de sélections

Accès aux enregistrements individuels

Chaque présentation contient en général un ou plusieurs objets Record (enregistrement). Cependant, s'il s'agit d'une nouvelle présentation, il est possible qu'elle ne contienne aucun enregistrement. Utilisez les propriétés des enregistrements de la classe Gallery pour itérer sur un ensemble d'enregistrements afin de manipuler les valeurs des champs contenus dans chaque enregistrement. Les enregistrements sont en général référencés dans l'objet `AllRecords`. Cet objet contient un objet Record de chaque enregistrement de la présentation sélectionnée. Notez bien qu'un index représente l'ordre actuel des enregistrements d'une présentation spécifique, par conséquent l'index d'un enregistrement donné variera selon le contenu et l'ordre de la présentation.

Pour utiliser un enregistrement spécifique d'un catalogue, transférez le numéro d'index de l'objet Record concerné vers l'objet `AllRecords`.

```
PortObj.Gallery(1).AllRecords(x)
```

Pour obtenir le décompte de tous les enregistrements d'une présentation, utilisez la propriété `Count`.

```
rCount = PortObj.Gallery(1).AllRecords.
Count
```

Assurez-vous que celle-ci vous renvoie le nombre d'enregistrements de la présentation, ce nombre n'étant pas forcément identique au nombre d'enregistrements du catalogue entier.

Utilisation de sélections

L'une des utilisations courantes des scripts (ou des programmes Visual Basic) consiste à exécuter des opérations selon les enregistrements sélectionnés par l'utilisateur dans le catalogue. Pour déterminer les enregistrements sélectionnés, référez-vous tout simplement à l'objet `SelectedRecords` d'un objet `Gallery` spécifique. Comme pour l'objet `AllRecords`, un index est utilisé pour identifier un objet Record spécifique dans l'objet `SelectedRecords`.

Par exemple :

```
PortObj.Gallery(1).SelectedRecords(x)
```

Comme pour l'objet `AllRecords`, `SelectedRecords` possède également une propriété Count qui renvoie le nombre d'enregistrements individuels sélectionnés. Vous pouvez alors utiliser cette propriété pour définir une boucle dans votre code Visual Basic et passer d'un enregistrement sélectionné à l'autre.

Sélection d'éléments

Pour modifier la sélection, utilisez les fonctions Select (sélectionner) et Deselect (désélectionner). La fonction Select ne déselectionne par la sélection actuelle, par conséquent vous pouvez avoir besoin d'utiliser au préalable la fonction Deselect pour annuler la sélection.

Exemples :

```
PortObj.Gallery(1).AllRecords(1).Select

PortObj.Gallery(1).SelectedRecords(2).
Deselect
```

De plus, l'objet `Gallery` possède une fonction `SelectAll`, qui définira le point de sélection de façon à inclure chaque enregistrement de la présentation.

Par exemple :

```
PortObj.Gallery(1).SelectAll
```

Il est possible que les modifications apportées à la sélection ne soient pas visibles à l'écran jusqu'à ce que vous ayez actualisé la présentation. La sélection sera correcte, mais il est possible que l'écran ne soit pas redessiné si les éléments manipulés sont déjà visibles.

L'affichage des listes ne prend pas en charge la modification de la sélection. L'objet `SelectedRecords` ***est accessible, mais les fonctions*** `Select` ***et*** `Deselect` ***ne fonctionneront pas.***

Utilisation de champs

Chaque objet `Record` contient un certain nombre d'objets Field (un par champ système, et un par champ personnalisé). Il est possible d'identifier un champ spécifique en transférant le nom de ce champ.

```
PortObj.Gallery(1).AllRecords(1).Field(x)
```

Pour déterminer la valeur d'un champ, utilisez la propriété Value de l'objet `Field`.

```
SDesc = PortObj.Gallery(1).AllRecords(1).
Field("Description").Value
```

Pour déterminer le nombre de champs d'un catalogue, exécutez la propriété `Count` d'un enregistrement spécifique.

```
fldCount = PortGal.AllRecords(1).Count
```

Détermination des champs d'un catalogue

La structure de données dynamique de Portfolio permet à l'utilisateur de personnaliser le catalogue en définissant des champs personnalisés. Etant donné que vous pouvez être amené à travailler avec une structure d'enregistrement contenant des champs de données personnalisés, il est souvent utile de déterminer quels champs sont définis dans un catalogue spécifique. Pour cela, parcourez la liste de tous les noms de champs d'un enregistrement spécifique (tous les enregistrements d'un même catalogue renverront les mêmes résultats), et examinez la propriété `Name`.

Par exemple :

```
fldCount = PortGal.AllRecords(1).Count
For i = 1 To fldCount
   If PortObj.Gallery(1).AllRecords(1).
   Fields(i).Name = "FieldName" Then
      rem Le champ nommé FieldName a été trouvé
   End If
Next
```

Modification des valeurs de champ

L'un des aspects les plus puissants de l'interface de script de Portfolio est sa capacité à non seulement lire tous les champs d'un enregistrement, mais aussi à modifier tous ces champs (y compris les champs système). N'oubliez pas que, bien que cet outil soit extrêmement utile, il est également possible d'endommager un catalogue en modifiant incorrectement des données utilisées par Portfolio pour gérer des fichiers source. Par exemple, si le chemin de chaque enregistrement d'un catalogue venait à être modifié incorrectement, il est possible que Portfolio ne puisse plus trouver d'enregistrements. Pour définir la valeur du champ, opérez de la même façon que pour obtenir les valeurs de champ.

Par exemple :

```
PortObj.Gallery(1).AllRecords(1).
Field("Description").Value = "This is the
new description"
```

Vous pouvez également trouver un exemple de code Visual Basic qui exécute une fonction Search and Replay (rechercher et rejouer) sur certains champs des tableaux de Portfolio, dans l'utilitaire **Path Change** fourni avec ce document.

Procédez avec prudence lorsque vous apprenez à manipuler le contenu des champs des tableaux de données de Portfolio. Il est possible d'effectuer des modifications de données qui pourraient corrompre l'intégralité de votre catalogue. Sauvegardez toujours vos données Portfolio avant d'essayer de modifier les données des tableaux à partir de votre code Visual Basic.

Transfert de valeurs de champ vers Portfolio

Types de données

Lorsque vous programmez avec les objets Portfolio, toutes les valeurs utilisées doivent être transférées vers Portfolio sous forme de chaînes, quel que soit le type de données du champ dans Portfolio. Si vous transférez des valeurs sous une forme autre qu'une variable chaîne, votre opération produira une erreur. Portfolio utilise ses propres routines de validation internes pour déterminer si les données transférées conviennent au type de champ, et faire en sorte que le processus interne n'accepte que les données de variables chaînes entrantes. L'exemple qui suit montre la définition du champ `Last Updated` (un champ de date) par le transfert d'une valeur chaîne.

Gardez ceci à l'esprit si votre application est destinée à utiliser une variable déclarée. Si vous avez par exemple une variable date, vous devez la convertir en chaîne avant de transférer cette valeur vers Portfolio.

Par exemple :

```
PortObj.Gallery(1).AllRecords(1).
Field("Last Updated").value = "July 4,
1776"
```

Champs à valeurs multiples

Les objets Field contiennent un ensemble de fonctions pour gérer les données à valeurs multiples. Pour lire la liste à valeurs multiples, utilisez la propriété `MVDataCount` afin de déterminer le nombre d'éléments dans la liste, puis itérez la liste à l'aide de la fonction `GetMVData` pour lire les valeurs.

Par exemple :

```
iCount = PortObj.Gallery(1).AllRecords(1).
Field("Keywords").MVDataCount

For i = 1 To iCount

   sValue = PortObj.Gallery(1).
   AllRecords(1).Field("Keywords").
   GetMVData(i)

   rem Traitent les données

Next i
```

Définissez la propriété `Value` de façon à ajouter une valeur à la liste à valeurs multiples. Utilisez la fonction `DeleteMVData` pour supprimer une valeur spécifique, ou utilisez la fonction `DeleteData` pour supprimer toutes les valeurs de la liste.

Exemples :

```
rem Ajoute Dog à la liste des mots-clés

PortObj.Gallery(1).AllRecords(1).
Field("Keywords").Value = "dog"

rem Supprime Dog de la liste des mots-clés

result = PortObj.Gallery(1).AllRecords(1).
Field("Keywords").DeleteMVData("dog")

rem Supprime tous les mots-clés de cet
élément

result = PortObj.Gallery(1).AllRecords(1).
Field("Keywords").DeleteData
```

Recherche dans le catalogue

Exécution d'une recherche

Pour effectuer une recherche dans un catalogue Portfolio, utilisez la fonction Find (rechercher) de l'objet `Gallery`. Les recherches effectuées dans Portfolio sont exécutées par la transmission d'une chaîne de texte représentant les critères de recherche. Le format de la chaîne de recherche doit correspondre au format utilisé dans la boîte de dialogue **Rechercher** de Portfolio. La fonctionnalité de base est montrée ci-dessous. Consultez la section suivante pour savoir comment formuler la variable de requête correctement.

```
Function Find(SearchString As String,
AllRecords As Boolean, SetNewGallery As
Boolean) As Integer
```

L'attribut `AllRecords` est l'équivalent de l'option **Rechercher dans la présentation** de la boîte de dialogue Rechercher de Portfolio. Une valeur `True` revient à ce que cette case soit décochée. L'attribut `SetNewGallery` est l'équivalent de l'option **Afficher les résultats dans une nouvelle présentation** de la boîte de dialogue Rechercher de Portfolio. Une valeur `True` revient à ce que cette case soit cochée.

Pour rechercher tous les enregistrements du catalogue, il vous suffit de transmettre une requête qui ne puisse pas échouer.

Par exemple :

```
PortObj.Gallery(1).Find(("Filename" &
vbTab & "starts with" & vbTab & ""),
True, False)
```

Formulation de la requête

Les recherches effectuées dans Portfolio sont exécutées par la transmission d'une chaîne de texte représentant les critères de recherche à un objet `Document`. La structure des informations de cette requête est présentée de la même façon que dans la boîte de dialogue **Rechercher** de Portfolio. Comme dans la boîte de dialogue **Rechercher** de Portfolio, la structure de requête de base consiste en trois clauses : le `field`, l'`operator`, et la `value`. Chacune de ces clauses est transmise sous la forme d'une chaîne de texte, et les clauses sont séparées par un caractère de tabulation.

Par exemple :

```
theQuery = "Keywords" & vbTab & "starts
with" & vbTab & "test"

PortObj.Gallery(1).Find(theQuery, True,
False)
```

Pour créer une recherche plus complexe, vous devez délimiter chaque ligne par un caractère de retour. De plus, chaque ligne après la première doit commencer par la condition de jointure (« et » ou « ou »). Vous trouverez ci-dessous un exemple de requête de recherche de deux lignes :

Exemple :

```
theQuery = "Filename" & vbTab & "starts
with" & vbTab & "test" & vbNewLine &
"and" & vbTab & "Keywords" & vbTab &
"starts with" & vbTab & "key"
```

Annexe A : Bibliothèques des classes de Portfolio

Cette annexe contient une liste classée par ordre alphabétique de tous les membres, fonctions, et propriétés de chacune des classes de Portfolio utilisées par le modèle d'automatisation.

Classe Document de Portfolio :

Count as Long

Gallery(nIndex as Long) as Object [read-only]

GetActive() as String

GetGalleryIndexFromName(GalleryName as String) as Integer

LastError as Long

New(Path as String) as Integer

NewCatalogWithPrompt() as Integer

Open(Path As String, AccessMode As Integer, Password As String) As Integer

OpenByUserName(Path As String, AccessMode As Integer, UserName As String, Password As String) As Integer

OpenServer(sAddress As String, AccessMode As Integer, Password As String) As Integer

OpenServerByUserName(sAddress As String, AccessMode As Integer, UserName As String, Password As String) As Integer

OpenServerWithPrompt(sAddress As String) As Integer

OpenWithPrompt(sPath As String) As Integer

SetActive(GalleryName As String) As Integer

Visible as Boolean

Classe Gallery de Portfolio :

Function **Activate**() As Integer

Function **AddFiles**(pDataObj As Unknown) As Boolean

Function **AddRecord**(szFile As String, szJPEGThumbnailFile As String, nThumbSize As Integer, nThumbQuality As Integer) As Object

AllRecords As Object

Busy as Long

Function **Catalog**(Path As String, IncludeDirs As Boolean, bUseUserSettings As Long) As Boolean

```
CatalogName as String

Function CatalogUsingOptionsPreset(Pa
th As String, szPresetName As String,
IncludeDirs As Boolean) As Boolean

Function Close() As Integer

Function CollectFiles(szFileCollec
tFolder As String, bKeepHierarchy
As Boolean, bCollectOriginals As
Boolean, nPreviewSizeInPixels As
Integer, bOrigIfNoPreview As Boolean,
bCreateArchiveCatalog As Boolean,
szCatalogPath As String, szVolumeName
As String, bOverwriteExistingCatalog
As Boolean, bIncludeBrowser As Boolean,
bIncludeUserGuide As Boolean) As Integer

Count as Long

Function CreateCustomField(szFieldNa
me As String, nFieldType As Integer,
bMultiValued As Boolean, nFieldOption As
Integer, szAdminPassword As String) As
Boolean

Function CreateCustomFieldPreDef(sz
FieldName As String, nFieldType As
Integer, bMultiValued As Boolean,
nFieldOption As Integer, bFromListOnly
As Boolean, szFieldValueList As String,
szAdminPassword As String) As Boolean

Function CreatePlaceholder(szFileName
As String, szOptionalPath As String) As
Long

Function DeleteRecord(RID As Long,
bDeleteFromCatalog As Boolean) As
Boolean

Function Find(SearchString As String,
AllRecords As Boolean, SetNewGallery As
Boolean) As Integer

GalleryName as String

Function GetRecordFromRecordID(nRecordID
As Long) As Object

Function ImportFieldValues(Path As
String, SavedSet As String) As Boolean

NumFound as Long

Options as Object

Sub RefreshView()

Function Save() As Integer

Function SaveAs(sNewName As String) As
Integer

Function SelectAll() As Integer

SelectedRecords As Object

Function Sort(FieldName As String,
Direction As Boolean) As Boolean

TotalRecordsInCatalog As Long
```

Classe Record de Portfolio :

Count as Long

Function **Delete**(bDeleteFromCatalog As Boolean) As Boolean

Deselect() as Boolean

Sub **EnsureVisible**()

Function **ExtractProperties**() as Boolean

Property **Field**(FieldName as String) as Object

Property **Fields**(nIndex as Long) as Object

Function **GetThumbnailData**(pData, nSize As Long) As Boolean

IsSelected as Boolean

RecordID as Long

Function **RegenerateThumbnail**() As Integer

Function **Select**() As Boolean

Function **Update**() As Boolean

Classe Field de Portfolio :

DeleteData() as Boolean

DeleteMVData(Value As String) As Boolean

GetMVData(nIndex As Long) As String

GetPredefinedValue(Index As Integer) As String

GetPredefinedValueCount() As Integer

GetPredefinedValueList() as String

IsCustom as Boolean

IsIndexed as Boolean

IsMultiValue as Boolean

IsPredefined as Boolean

IsURL as Boolean

MyDataCount as Long

Name as String

Type as Integer

Value as String

Classe Records de Portfolio :

Count as Long

Function **GalleryCopy**(nGalleryIndex as Long)as Integer

Property **Records**(nIndex as Long) as Object [read-only]

Classe Selection de Portfolio :

Count as Long

Function **GalleryCopy**(nGalleryIndex as Long)as Integer

Property **Records**(nIndex as Long) As Object [read-only]

Classe AdvancedOptions de Portfolio :

AppendDescription as Integer

ExtractMetadata as Boolean

ExtractThumbnail as Boolean

MergeKeywords as Integer

ParseKeywords as Integer

PathKeywords as Integer

SkipFiles as Boolean

ThumbnailSize as Integer

ANNEXE B : Questions fréquentes et réponses

Cette annexe contient certaines des questions les plus fréquentes sur la programmation VB avec l'objet document de Portfolio.

Comment faire pour ajouter les références Portfolio à mon projet Visual Basic ?

Dans le menu **Projet**, sélectionnez l'option références. Localiser la référence de la bibliothèque Portfolio_V8, sélectionnez-la et cliquez sur le bouton **OK**. L'objet document de Portfolio sera désormais inclus dans votre projet source VB, et ses objets et propriétés prêts à être utilisés dans votre code VB. Reportez-vous au début du didacticiel pour obtenir des exemples du code requis pour déclarer et définir un objet document de Portfolio dans votre projet.

Puis-je utiliser VBA ou VB Script avec Portfolio, ou dois-je utiliser la version complète de Visual Basic ?

La façon dont les objets sont définis et gérés est très différente selon qu'il s'agit de la version complète de Visual Basic ou de VBA et VB Scripting. Du fait de ces différences, Portfolio ne prend en charge que l'utilisation de la version complète de Visual Basic ou Visual Basic.net.

Qu'est-ce que la variable PortObj à laquelle vous faites souvent allusion ?

Dans le code exemple, et dans ce document, nous utilisons le nom de variable `PortObj` pour identifier l'objet document de Portfolio auquel il est fait référence dans le code. Vous êtes libre de choisir le nom de variable de votre choix dans votre propre code, mais dans un but purement explicatif, nous utilisons ce nom standard d'objet dans toute la documentation et dans le code exemple.

Puis-je utiliser la bibliothèque d'automatisation de Portfolio avec VB.Net ?

Le modèle objet d'automatisation utilisé par Portfolio fonctionnera avec VB.net, mais vous ne devez pas oublier que si vous développez une application en utilisant l'objet d'automatisation de Portfolio sous VB.net, vous devrez installer Net Framework sur tous les ordinateurs qui exécuteront votre application.

Où puis-je trouver les définitions de champs pour les tableaux de Portfolio ?

Sous Portfolio, utilisez l'option **Personnaliser l'affichage** pour afficher la liste complète des champs inclus dans un catalogue. Vous pouvez utilisez ceci pour sélectionner les champs que vous souhaitez voir affichés. Utilisez ceci comme outil de débogage lorsque vous écrivez du code pour éditer ou modifier des valeurs de champ. Cette opération vous permet de voir immédiatement si votre code a exécuté les modifications que vous souhaitiez apporter aux valeurs de champ.

Si vous souhaitez lire ou afficher la liste des champs disponibles depuis votre application Visual Basic, reportez-vous à la section **Utilisation de champs** du didacticiel.

Comment puis-je connaître le nombre de présentations ouvertes ?

Vous pouvez obtenir un décompte du nombre de présentations actuellement ouvertes grâce au code suivant :

```
PortObj.Document.Count
```

Notez bien que la valeur de cette propriété Count renverra le nombre de toutes les présentations actuellement ouvertes dans tous les catalogues ouverts. Par exemple, si deux catalogues sont ouverts, que chacun compte trois présentations, et qu'une présentation est ouverte dans chaque catalogue, la propriété Count renverra une valeur de deux.

Puis-je activer une présentation spécifique à partir de mon code VB ?

Oui. Utilisez la commande suivante :

```
PortObj.SetActive("CatalogName.fdb -
GalleryName")
```

N'oubliez pas que vous devez utiliser le nom complet de la présentation, y compris le catalogue parent. Vous ne pouvez pas utiliser seulement le nom de la présentation. Vous pouvez trouver le nom complet de la présentation sous Portfolio en regardant la barre de légende de la fenêtre de la présentation.

Comment puis-je obtenir le nom du catalogue ?

Comme vous l'avez sans doute remarqué, sans classe ou objet Catalog, il n'existe aucune façon évidente de lire le nom du catalogue. Souvenez-vous, l'objet document est en fait une collection de présentations, et un objet `Gallery` possède une propriété `Catalog Name`. Vous pouvez utiliser la propriété suivante pour lire le nom du catalogue actuel :

```
NameVariable = PortGal.CatalogName
```

Vous pouvez également référencer le nom du catalogue en vous intéressant à la propriété Gallery. Name qui inclura le nom du catalogue comme faisant partie de la définition du nom de la présentation.

Par exemple :

```
Gallery Name: ("MasterCatalog.fdb -
Sample Gallery")
```

Lorsque j'essaie de définir des valeurs de champ, j'obtiens toujours des erreurs. Comment cela se fait-il ?

Reportez-vous à la section du didacticiel concernant le **Transfert de valeurs de champ vers Portfolio**. Le plus souvent, cette erreur est due au fait que les valeurs que vous transférez vers l'objet Portfolio ne sont pas des chaînes. Toutes les valeurs de champ, quel que soit leur type, doivent être transférées sous forme de valeurs chaînes.

ANNEXE C : Guide d'utilisation des exemples d'applications

Démonstration de l'utilisation de l'interface d'automatisation de Portfolio :

Cette application fournit de nombreuses fonctions utilitaires, mais pour un programmeur VB, son plus grand atout demeure la vaste gamme d'objets de classe et de méthodes qu'elle utilise. En parcourant le code exécuté, vous pouvez suivre la plupart des principales fonctions de programmation qui sont à votre disposition tout en apprenant à programmer à l'aide de l'objet document de Portfolio et de ses classes.

Utilitaire Path Change (pour modifier le chemin d'accès) :

C'est un utilitaire très utile qui vous permet de modifier les définitions de chemin d'accès dans la base de données de Portfolio si vous avez changé des images cataloguées d'emplacement.

Vous ne devez pas oublier que deux champs stockent les informations relatives au chemin d'accès à un enregistrement (`Path` et `Directory Path`) et que par conséquent, si vous souhaitez redéfinir les informations relatives au chemin d'accès à un enregistrement, vous devez modifier ces deux champs.

Vous pouvez également examiner le code pour trouver un exemple sur la façon de lire et écrire des valeurs de champ à partir de votre code VB.

Utilitaire de sauvegarde :

Il s'agit d'un simple programme utilitaire qui vous permet de sauvegarder un catalogue Portfolio existant. Bien qu'il n'utilise aucune des méthodes d'automatisation de Portfolio, cet utilitaire fournit un exemple sur la façon d'arrêter et démarrer le serveur de Portfolio à partir de votre code VB en utilisant des fichiers `.BAT` prédéfinis.

ANNEXE D : Codes d'erreur de programmation de Porfolio

Cette annexe contient une liste classée par ordre alphabétique de tous les membres de chaque objet de classe de Portfolio.

Lorsqu'une erreur se produit, vous pouvez obtenir la valeur du code d'erreur Portfolio en lisant la propriété `ObjectName.LastError`. Chaque fois qu'une fonction ou un appel est exécuté avec succès, la valeur de cette erreur est définie sur zéro.

Utilisez les procédures de détection d'erreurs Visual Basic standard pour tester et gérer ces erreurs. (Reportez-vous à vos manuels de référence VB si vous avez des doutes quant à la gestion des erreurs dans votre application VB.)

KScriptNoError	**0**	**Aucune erreur (succès)**
KScriptBadPassword	**1**	**Mot de passe incorrect ou non valide**
KScriptCatInUse	**2**	**Le catalogue est en cours d'utilisation et ne peut donc pas être ouvert sous ce mode. Ce code d'erreur n'est pas utilisé dans le code client Portfolio ; vous ne devriez donc pas le rencontrer.**
KScriptAdminCatInUse	**3**	**Tente d'ouvrir un catalogue en mode Admin alors qu'il existe d'autres utilisateurs**
KScriptAdminInUse	**4**	**Un autre utilisateur est déjà connecté en tant qu'administrateur**
KScriptReadOnlyAvailable	**5**	**Le catalogue n'est disponible qu'en lecture seule, sur un volume en lecture seule par exemple**
KScriptServedAdminError	**6**	**Ouvre un catalogue servi en mode Admin (ou en mode Navigateur)**
KScriptUnknownCatError	**7**	**Erreur inconnue lors de l'ouverture du catalogue**
KScriptInvalidMode	**8**	**Mode non valide**
KScriptNoFileName	**9**	**Aucun nom de fichier**
KScriptDiskFull	**10**	**Disque saturé**
KScriptErrorCreate	**11**	**Erreur lors de la création d'un catalogue**
kScriptInvalidFieldname	**12**	**Nom de fichier incorrect**
kScriptFieldNotIndex	**13**	**Le champ n'est pas indexé**

kScriptFieldNotMultiVal	14	Ce n'est pas un champ à valeurs multiples
kScriptIndexOutOfRange	15	Index hors limites
kScriptEmptyValue	16	Valeur vide
kScriptInvalidValueforFieldType	17	La valeur n'est pas valide pour le type de champ donné
kScriptValueNotInPredefList	18	La valeur ne se trouve pas dans la liste prédéfinie
kScriptCatalogNotFound	19	Catalogue introuvable
kScriptServerNotFound	20	Serveur introuvable
kScriptMaxServerConnected	21	Le nombre maximum autorisé de connexions au serveur a été atteint
kScriptInvalidFieldValue1	22	La valeur (FieldValue) du champ 1 n'est pas valide
kScriptInvalidFieldValue2	23	La valeur (FieldValue) du champ 2 n'est pas valide
kScriptInvalidGallery	24	Présentation non valide
kScriptInvalidQuery	25	La requête (Rechercher) n'est pas correctement spécifiée
kScriptSystemField	26	Impossible de modifier le champ système (intégré)
kScriptGalleryNotFound	27	Présentation introuvable
kScriptFieldNotPredefined	28	Le champ n'est pas de type prédéfini
kScriptInvalidCatalogOption	29	Option de catalogage non valide
kScriptInvalidHTMLSetName	30	Le nom défini enregistré en langage HTML n'est pas valide
kScriptCreatingHTMLImagesDir	31	Erreur lors de la création d'un répertoire d'images HTML
kScriptCreatingHTMLFile	32	Erreur lors de la création d'un fichier HTML
kScriptWritingHTMLFile	33	Erreur lors de l'écriture vers un fichier HTML
kScriptOutOfMemory	34	Mémoire insuffisante
kScriptCopyError	35	Erreur de copie (GalleryCopy)
kScriptServerDisconnect	36	Serveur déconnecté
kScriptInvalidForFoundItems	37	Impossible d'exécuter l'opération sur la présentation des « éléments trouvés »
kScriptInvalidLoginMode	38	Tente d'ouvrir un catalogue accessible en fonction de l'utilisateur, avec une interface API basée sur le niveau
kScriptUnknownError	1000	Erreur inconnue lors de l'ouverture ou la création d'un catalogue