macropool

ContentSaver

Programmierhandbuch

Inhaltsverzeichnis

Einführung

Dieses Kapitel gibt Ihnen eine Einführung zur Verwendung der ContentSaver-Programmierschnittstelle. Es werden folgende Themen behandelt:

- Zweck der ContentSaver-Programmierschnittstelle
- Anwendungsbeispiele
- Voraussetzungen
- Die ContentSaver-Objektbibliothek

Zweck der Programmierschnittstelle

Für viele Anwender ist ContentSaver inzwischen zum unverzichtbaren Werkzeug für die Sammlung und Verwaltung jeglicher Art von Information geworden. Nicht nur Inhalte aus dem Internet, sondern auch beliebige andere Informationen wie z.B. auf der Festplatte bereits vorhandene Dokumente, E-Mails oder Dateien lassen sich mit ContentSaver sammeln und professionell strukturieren. Im Laufe der Zeit bilden sich so umfangreiche Informationssammlungen, die für die Anwender von großem Wert sind. Sie wünschen sich daher umso mehr, dass diese Wissensbasis vielseitig und flexibel verwendbar ist und nicht in einer abgeschotteten Softwarelösung gespeichert bleibt.

Deshalb bietet ContentSaver eine frei programmierbare Schnittstelle, die einen Zugriff auf die ContentSaver-Archive, -Ordner und -Dokumente ermöglicht. Sie können damit die Eigenschaften von Dokumenten auslesen oder ändern sowie neue Dokumente hinzufügen. Ebenso lassen sich Ordner hinzufügen, löschen, verschieben oder umbenennen. Selbst die Kategorien sind änderbar und können Dokumenten programmiert zugewiesen werden. Damit sind die vielfältigsten Einsatzmöglichkeiten denkbar!

Anwendungsbeispiele

Mit der Programmierbarkeit der ContentSaver-Archive eröffnen sich bislang ungeahnte Verwendungsmöglichkeiten für ContentSaver.

Automation von Aktionen in ContentSaver

Sie können Makros programmieren, mit denen Sie in großen Archiven automatisiert Objekte und deren Eigenschaften ändern. Denkbar wären z.B. folgende Anwendungen:

- Ein eigener Exportfilter, der Daten eines bestimmten Ordners in gewünschter Form in eine Datei schreibt oder an eine andere Anwendung übergibt.
- Automatisierte Umkategorisierung von Dokumenten
- Speicherung eigener Dokumente und Daten, ohne dass ein Benutzerinterface (Internet Explorer bzw. ContentSaver-Anwendung) dazu notwendig wäre.

Hinweis: Im Gegensatz zu Microsoft Outlook ist es nicht möglich, die Benutzeroberfläche der ContentSaver-Anwendung zu automatisieren. Die Programmierschnittstelle von ContentSaver dient allein dem Zugriff auf ContentSaver-Archive, nicht auf die ContentSaver-Anwendung.

Einbindung in andere Anwendungen

Über die Programmierschnittstelle können Sie in anderen Anwendungen neue Dokumente zu ContentSaver-Archiven hinzufügen oder bereits vorhandene ContentSaver-Dokumente abrufen. Externe Anwendungen können somit Daten in die Archive einspeisen oder daraus auslesen. Im Beispielabschnitt dieser Dokumentation wird ein kleines Programm vorgestellt, wie Sie in Microsoft Outlook die aktuell angezeigte E-Mail-Nachricht in ContentSaver speichern können.

Programmieren eigener Anwendungen

ContentSaver ist derart flexibel, dass Sie damit beliebige eigene Informationen speichern können. Es muss sich hierbei nicht zwingend um Daten handeln, die mit ContentSaver im Explorer gespeichert werden. Sie können auch selbst generierte Daten in Dokumenten ablegen und diese mittels Ordnern und Kategorien strukturieren.

Voraussetzungen

Die Datenschnittstelle ist in ContentSaver in Form einer ActiveX (COM)-Komponente realisiert. Hierbei handelt es sich um eine bewährte Technologie, die von den meisten Programmiersprachen unterstützt wird. Dazu gehören die Microsoft-Entwicklungssysteme Visual Basic (VB), Visual Basic für Applikationen (VBA), VBScript und Visual C++. Aber auch mit vielen Programmiersprachen anderer Hersteller können ActiveX-Komponenten genutzt werden, z.B. mit Borland Delphi.

Der programmierte Zugriff auf die ContentSaver-Daten erfolgt über Objekte, die die einzelnen Elemente im Archiv repräsentieren. Deshalb trägt das Softwaremodul, das die ContentSaver-Programmierschnittstelle realisiert, die Bezeichnung *ContentSaver-Objektbibliothek*.

Im nächsten Abschnitt werden sämtliche Objekte und deren Programmierung anhand von kleinen Visual Basic-Codebeispielen dargestellt. Um die in dieser Beschreibung behandelten Beispiele auszuprobieren, benötigen Sie daher einen Visual Basic-Editor. Darüber hinaus sollten sie über grundlegende Kenntnisse der Programmierung in Visual Basic verfügen.

Sofern Sie auf Ihrem Rechner ein Microsoft Office-Produkt installiert haben, besitzen Sie mit hoher Wahrscheinlichkeit bereits einen Visual Basic-Editor, denn die meisten Microsoft Office-Anwendungen haben ab der Version 97 einen solchen integriert. Damit können Makros bearbeitet oder eigene VBA-Programme entwickelt werden.

Die ContentSaver-Objektbibliothek

Allgemeine Informationen

Sämtliche Elemente in ContentSaver wie z.B. Archive, Ordner oder Dokumente werden zum Zweck der Programmierung zu *Objekten* abstrahiert. Diese Objekte haben wiederum *Eigenschaften* und *Methoden*, welche die einzelnen Aktionen ausdrücken, die man mit ihnen durchführen kann. Die Gesamtheit aller Objekte, deren Eigenschaften und Methoden, sowie deren Beziehungen untereinander, wird gemeinhin in Form eines *Objektmodells* beschrieben.

Bevor Sie innerhalb Ihrer Visual Basic-Programmierumgebung ContentSaver-Objekte erzeugen, deren Methoden verwenden und ContentSaver-Konstanten verwenden können, müssen Sie eine Referenz zur ContentSaver-Objektbibliothek herstellen. Dies wird im folgenden Abschnitt erläutert.

Einbinden in Visual Basic für Applikationen (VBA)

Starten Sie z.B. Microsoft Outlook oder Microsoft Word. Zeigen Sie im Menü **Extras** auf **Makro,** und klicken Sie dann auf **Visual Basic-Editor**. Der Visual Basic-Editor öffnet sich. Wählen Sie nun das Projekt aus, in dem Sie die ContentSaver-Programme speichern wollen, und fügen ein neues Modul hinzu. Klicken Sie im Menü **Extras** auf **Verweise...**, und wählen Sie dann in der Liste der verfügbaren Verweise den Eintrag **ContentSaver Objektbibliothek** aus (dahinter steht noch eine Versionsnummer, z.B. „ContentSaver Objektbibliothek 1.5“).

Aufgrund der Risiken von Makroviren sind Microsoft Office-Anwendungen bei der Ausführung von Makros oder VBA-Programmen häufig sehr restriktiv. Bevor Sie die vorgestellten Beispiele ausprobieren können, müssen Sie unter Umständen erst die Sicherheitsstufe für die Ausführung von Makros in Ihrer Office-Anwendung ändern.

Einbinden in Visual Basic (VB)

Starten Sie Visual Basic und legen Sie ein neues **Standard-EXE-Projekt** an. Anschließend fügen Sie ein neues Modul, welches Sie z.B. "modContentSaver" benennen, hinzu.

Im nächsten Schritt müssen Sie einen Verweis zur ContentSaver-Objektbibliothek herstellen. Klicken Sie dazu im Menü **Projekt** auf **Verweise**, und wählen Sie in der Liste der verfügbaren Verweise den Eintrag **ContentSaver Objektbibliothek** aus (dahinter steht noch eine Versionsnummer, z.B. „ContentSaver Objektbibliothek 1.5“).

Das ContentSaver-Objektmodell

In diesem Kapitel werden alle Objekte der ContentSaver-Programmierschnittstelle dargestellt und mit kleinen Code-Beispielen erläutert.

Das Namespace-Objekt

Das `Namespace`-Objekt ist das oberste Element im ContentSaver-Objektmodell. Ohne `Namespace`-Objekt ist kein Zugriff auf die Archive möglich. Dieses Objekt verfügt über folgende Eigenschaften und Methoden:

Eigenschaften des Namespace-Objektes

Name	Beschreibung	Datentyp	R/W
`Archives`	Gibt eine Auflistung aller in ContentSaver geöffneten Archive zurück.	`Collection/ Archive`	R
`CurrentUser`	Name des auf dem lokalen Rechner angemeldeten Benutzers.	`String`	R
`TempPath`	Ordnerpfad, den ContentSaver für temporäre Dateien verwendet.	`String`	R
`Progress`	Setzt das Fenster-Handle einer Progress Bar zur Anzeige eines Fortschrittsbalkens bei längeren Lösch-, Kopier- oder Verschiebeoperationen	`Long`	R/W
`DefaultArchive`	Setzt das Standardarchiv oder gibt es zurück.	`Object/ Archive`	R/W

So erzeugen Sie ein Namespace-Objekt:

```
' Namespace-Variable deklarieren
Dim csNS as CSObj.Namespace
' Referenz zu einem ContentSaver-Namespace Objekt erzeugen
Set csNS = New CSObj.Namespace
```

Hinweis: Sämtliche nachfolgenden Code-Beispiele setzen diese beiden Anweisungen voraus! Sie werden aus Gründen der Übersichtlichkeit nicht mehr wiederholt.

So sprechen Sie Ihr Standardarchiv an:

```
' Archive-Variable deklarieren
Dim csArc as CSObj.Archive

' Referenz auf Standardarchiv holen
Set csArc = csNS.DefaultArchive
```

Methoden des Namespace-Objektes

Name	Beschreibung	Parameter
`RemoveTempFiles`	Löscht alle temporären Dateien einer ContentSaver-Sitzung	

Zum Anzeigen der Dokumente im Vorschaufenster schreibt ContentSaver alle Dokumente in einen temporären Ordner, der mit der `TempPath`-Eigenschaft

ermittelt werden kann. Beim Beenden von ContentSaver werden diese Dokumente mit der `RemoveTempFiles`–Methode wieder gelöscht.

Die Archives-Auflistung

Die `Archives`-Eigenschaft des `Namespace`-Objektes liefert eine Auflistung aller in ContentSaver geöffneten Archive. Haben Sie nur ein Archiv, so enthält diese Auflistung nur ein Element. Der Zugriff auf ein einzelnes Archiv ist über die `Item`-Eigenschaft möglich.

Eigenschaften der Archives-Auflistung

Name	Beschreibung	Datentyp	R/W
`Count`	Anzahl der in ContentSaver geöffneten Archive	`Long`	R
`Item`	Gibt ein Archiv-Objekt aus der Auflistung zurück. Ein selektiver Zugriff ist über die `ArchiveID` als Key möglich.	`Object/ Archive`	R
`Namespace`	Gibt das `Namespace`-Objekt zurück.	`Object/ Namespace`	R

So greifen Sie auf alle verfügbaren Archive zu:

```
' Archives-Variable deklarieren
Dim csArcs as CSObj.Archives
' Archive-Variable deklarieren
Dim csArc as CSObj.Archive

' Referenz auf alle ContentSaver-Archive holen
Set csArcs = csNS.Archives
' Alle Archive durchgehen und deren Namen ausgeben
For Each csArc in csArcs
   Debug.Print csArc.Name
Next
' Auf das erste Archiv in der Auflistung zugreifen
Set csArc = csArcs.Item(1)
' Ein Archiv mit der ArchiveID 2 explizit ansprechen
' Wichtig: Die ArchiveID muss der Item-Eigenschaft
' als String übergeben werden
Set csArc = csArcs.Item("2")
```

Methoden der Archives-Auflistung

Name	Beschreibung	Parameter
`Add`	Fügt ein Archiv in ContentSaver hinzu.	Name (String), ConnectionString (String)
`Remove`	Entfernt ein Archiv aus der Auflistung im `Namespace`-Objekt.	ArchiveID (String) *oder* Index (Long)

Das Archive-Objekt

Das `Archive`-Objekt repräsentiert ein einzelnes ContentSaver-Archiv. Ausgehend von diesem Objekt erhalten Sie alle Ordner und Kategorien des Archivs.

Eigenschaften des Archive-Objektes

Name	Beschreibung	Datentyp	R/W

ArchiveID	Laufende Nummer des Archivs. Diese ist einmalig und wird automatisch vergeben.	Long	R
Categories	Gibt eine Auflistung aller Kategorien im Archiv zurück.	Collection/ Categories	R
Comment	Kommentar zum Archiv	String	R/W
Fields	Gibt eine Auflistung aller Zusatzfelder des Archivs zurück.	Collection/ Fields	R
Folders	Gibt eine Auflistung aller Stammordner im Hauptverzeichnis des Archivs zurück.	Collection/ Folders	R
Location	Pfad zur Archivdatei	String	R
Name	Name des Archivs	String	R/W
Namespace	Gibt das übergeordnete ContentSaver-Namespace-Objekt zurück.	Object/ Namespace	R

Das Archive-Objekt besitzt eine Reihe von Methoden, mit denen Objekte direkt über deren ID angesprochen werden können. Sie sind daher besonders effizient, da keine Auflistung gebildet und durchsucht werden muss. Beispiele zur Verwendung dieser Methoden finden Sie in den folgenden Abschnitten.

Methoden des Archive-Objektes

Name	Beschreibung	Parameter
AddDocument	Fügt dem Archiv ein neues Dokument hinzu.	URL, DocType, Title, Content
DiscardSearch-Results	Die Suchergebnisse im Ordner *Suchergebnisse* werden verworfen.	
DocCountIn-Folder	Hocheffiziente Methode, um die Anzahl der Dokumente in einem Ordner zu ermitteln.	FolderID (Long)
GetCategory-FromID	Gibt ein Kategorie-Objekt anhand der CategoryID zurück.	CategoryID (Long)
GetDocFromID	Gibt ein Dokument-Objekt anhand der EntryID des Dokuments zurück.	EntryID (Long)
GetFolder-FromID	Gibt ein Ordner-Objekt anhand der FolderID des Ordners zurück.	FolderID (Long)
GetFolderFromPath	Gibt ein Ordner-Objekt anhand des Pfades des Ordners zurück.	Path (String)
NotifyCSApp	Benachrichtigt eine laufende ContentSaver-Anwendung, dass die Ansicht im angegebenen Ordner aktualisiert werden soll.	FolderID (Long)
SearchDocByUrl	Sucht ein Dokument anhand von Adresse und Größe im gesamten Archiv.	Url (String), *FileSize (String), *EntryID (Long)
WriteDocument	Schreibt ein Dokument in den temporären ContentSaver-Ordner auf die Festplatte.	EntryID (Long)

Die Folders-Auflistung

Ordner sind im ContentSaver-Objektmodell hierarchisch angeordnet. Aus diesem Grund werden alle Ordner innerhalb einer Ordnerebene in einer Auflistung zusammengefasst. Jeder Ordner, in dem Unterordner vorhanden sind, hat weitere Elemente in seiner `Folders`-Auflistung. Auf diese Weise können Sie sich durch einen Ordnerbaum hindurcharbeiten. Einzelne Ordner können innerhalb der Auflistung mit ihren Namen über die `Item`-Eigenschaft angesprochen werden.

Eigenschaften der Folders-Auflistung

Name	Beschreibung	Datentyp	R/W
`Count`	Anzahl der Ordner innerhalb der Auflistung	`Long`	R
`Item`	Ermöglicht ein Zugriff auf einen einzelnen Ordner. Ein selektiver Zugriff ist über den Namen möglich.	`Object/ Folder`	R
`Parent`	Gibt den übergeordneten Ordner zurück.	`Object/ Folder`	R

So greifen Sie auf einen Stammordner zu:

```
' Folders-Variable deklarieren
Dim csFldr as CSObj.Folder
' Referenz auf den Stammordner "Firma"
Set csFldr = arc.Folders.Item("Firma")
```

Wenn die `FolderID` eines Ordners bekannt ist, empfiehlt sich die Verwendung der Methode `GetFolderFromID` des Archiv-Objektes. Alternativ können Sie auch die Methode `GetFoldeFromPath` verwenden, die einen speziellen Ordner über den Pfad zurückgibt.

So greifen Sie auf einen Ordner über dessen ID zu:

```
Dim lFolderID As Long
' FolderID vom obigen Ordner merken
lFolderID = csFldr.FolderID
' Referenz auf den Ordner über die FolderID holen
Set csFldr = arc.GetFolderFromID(lFolderID)
```

Methoden der Folders-Auflistung

Name	Beschreibung	Parameter
`Add`	Fügt der Auflistung einen neuen Ordner hinzu.	Name (String)

So legen Sie einen neuen Stammordner an:

```
' Folders-Variable deklarieren
Dim csFldrs as CSObj.Folders
' Folder-Variable deklarieren
Dim csFldr as CSObj.Folder
' Referenz auf alle Stammordner des Archivs holen
Set csFldrs = arc.Folders
' Neuen Stammordner hinzufügen
Set csFldr = csFldrs.Add("ContentSaver")
```

So fügen sie einen Unterordner hinzu:

```
' Zusätzliche Variable für den Unterordner deklarieren
Dim csFldr2 as CSObj.Folder
```

```
' Neuen Unterordner hinzufügen
Set csFldr2 = csFldr.Add("Dokumentation")
```

Das Folder-Objekt

Das `Folder`-Objekt repräsentiert einen einzelnen Ordner im Archiv.

Eigenschaften des Folder-Objekts

Name	Beschreibung	Datentyp	R/W
`Archive`	Gibt das Archiv-Objekt zurück, innerhalb dessen sich der Ordner befindet.	`Object/ Archive`	R
`Comment`	Kommentar zum Ordner	`String`	R/W
`CreationTime`	Datum und Zeit, wann der Ordner angelegt wurde.	`Date`	R/W
`Documents`	Gibt eines Auflistung aller Dokumente des Ordners zurück.	`Collection/ Documents`	R
`Fields`	Gibt eine Auflistung aller Zusatzfelder des Ordners zurück.	`Collection/ Fields`	R
`FolderID`	Eindeutige Nummer des Ordners im Archiv	`Long`	R
`Folders`	Gibt eine Auflistung aller Unterordner zurück.	`Collection/ Folders`	R
`Name`	Name des Ordners	`String`	R/W
`Parent`	Gibt das übergeordnete Ordner-Objekt zurück.	`Object/ Folder`	R

So greifen Sie auf einen bestimmten Ordner zu:

```
' Folder-Variable deklarieren
Dim csFldr as CSObj.Folder
' Zugriff auf beliebige Ordner mittels Pfad
Set csFldr = csArc.GetFolderFromPath("Hobbies\Modellbau")
```

Sie können auch mithilfe von Konstanten auf häufig benutzte Standardordner zugreifen. Die untere Tabelle beschreibt diese Konstanten.

```
' Zugriff auf den Ordner „Neue Dokumente“
Set csFldr = csArc.GetFolderFromID(csFolderNewDocuments)
```

Ordnerkonstanten für Standardordner

Ordner	Konstante	Wert
Wurzelordner	`csFolderRoot`	1
Schablonen	`csFolderTemplates`	100
Kategorien	`csFolderCategories`	110
Suchvorlagen	`csFolderSearchTemplates`	140
Kontakte	`csFolderContacts`	150
Ideen	`csFolderIdeas`	200
Notierte Internetadressen	`csFolderCollectedLinks`	300

Neue Dokumente	csFolderNewDocuments	400
Suchergebnisse	csFolderSearchResults	500
Gelöschte Dokumente	csFolderDeletedDocuments	600
Dokumenteneingang	csFolderReceivedDocuments	700

So benennen Sie einen Ordner um:

```
' Zugriff auf einen beliebigen Ordner mittels Pfad
Set csFldr = csArc.GetFolderFromPath("Daten\Arbeitsamt")
' Umbenennen und speichern
csFldr.Name = "Arbeitsagentur"
csFldr.Update
```

Methoden des Folder-Objekts

Name	Beschreibung	Parameter
Copy	Kopiert den Ordner in einen anderen Ordner oder in ein anderes Archiv.	DestFolder (Folder)
Delete	Verschiebt den Ordner in den Ordner "Gelöschte Dokumente" oder löscht ihn endgültig.	Final (Boolean)
Move	Verschiebt den Ordner in einen anderen Ordner oder in ein anderes Archiv.	DestFolder (Folder)
Update	Speichert Änderungen an den Eigenschaften des Ordners.	

Die Methoden Copy, Delete und Move wirken sich immer auch auf alle Dokumente und Unterordner aus, die sich innerhalb des Ordners befinden.

So verschieben Sie einen Ordner in einen anderen Ordner:

```
' Folder-Variable für den Zielordner deklarieren
Dim DestFld as CSObj.Folder
' Zunächst eine Referenz auf den Zielordner holen
Set DestFld = arc.GetFolderFromPath("Umfrage\Auswertung")
' Ordner verschieben
fld.Move DestFld
' Ordner kopieren
fld.Copy DestFld
```

Hinweis: Sämtliche Änderungen an Standardordnern (z.B. "Neue Dokumente" oder "Dokumenteneingang") bleiben wirkungslos.

Die Documents-Auflistung

Jedes `Folder`-Objekt besitzt die `Documents`-Eigenschaft, mit der Sie eine Auflistung aller Dokumente dieses Ordners erhalten.

Eigenschaften der Documents-Auflistung

Name	Beschreibung	Datentyp	R/W
`Count`	Anzahl der Dokumente innerhalb der Auflistung	`Long`	R
`Item`	Ermöglicht den Zugriff auf ein einzelnes Dokument in der Auflistung. Zugriff über die `EntryID` möglich.	`Object/ Document`	R
`Parent`	Gibt den übergeordneten Ordner zurück.	`Object/ Folder`	R

So navigieren Sie durch alle Dokumente eines Ordners:

```
' Document-Variablen deklarieren
Dim csDocs as CSObj.Documents
Dim csDoc as CSObj.Document
' Auflistung aller Dokumente des Ordners bilden
Set csDocs = csFldr.Documents
For Each csDoc In csDocs
   ' Adresse ausgeben
   Debug.Print csDoc.Url
Next
```

Das Document-Objekt

Ein gespeichertes Element wird in ContentSaver durch das `Document`-Objekt repräsentiert. Sie erhalten ein einzelnes Dokument, indem Sie es aus der `Documents`-Auflistung eines Ordners herausnehmen oder mit der Methode `GetDocFromID` anhand der `EntryID` abrufen.

Eigenschaften des Document-Objekts

Name	Beschreibung	Datentyp	R/W
`Categories`	Gibt eine Auflistung aller dem Dokument zugewiesenen Kategorien zurück.	`Object/ Categories`	R
`CodePage`	Windows-Zeichensatz des Dokuments	`Long`	R/W
`Comment`	Kommentar zum Dokument	`String`	R/W
`Content`	Inhalt des Dokuments	`String`	R/W
`CreationTime`	Erzeugungszeitpunkt des Dokuments	`Date`	R/W
`DocType`	Dokumenttyp	`Long/ csDocType`	R/W
`EntryID`	Eindeutige ID des Dokuments innerhalb des Archivs	`Long`	R
`Fields`	Gibt eine Auflistung aller Zusatzfelder des Dokuments zurück.	`Object/ Fields`	R
`Filename`	Dateiname des Dokuments. Mit diesem wird das Dokument auf Datenträger	`String`	R/W

	geschrieben.		
FolderID	FolderID des Ordners, in dem das Dokument enthalten ist.	Long	R
Importance	Wichtigkeit des Dokuments	Long/ csDocImportance	R/W
LastModified	Datum der letzten Änderung des Dokuments	Date	R/W
Parent	Gibt das übergeordnete Objekt zurück. Dies ist meist ein Ordner.	Object/ Folder	R
Size	Größe des Dokuments in Byte	Long	R/W
Title	Titel des Dokuments	String	R/W
Url	Ursprungsadresse des Dokuments	String	R/W

So ändern Sie den Titel eines Dokuments:

```
' Document-Variable deklarieren
Dim csDoc as CSObj.Document
' Referenz auf Dokument mit EntryID holen
Set csDoc = csArc.GetDocFromID(lEntryID)
' Titel ändern und speichern
csDoc.Title = "ContentSaver-Dokumentation"
csDoc.Update
```

Sämtliche Änderungen an einem Dokument werden erst mit dem Aufruf der Update-Methode im Archiv gespeichert.

Methoden des Document-Objektes

Name	Beschreibung	Parameter
Copy	Kopiert ein Dokument in einen anderen Ordner oder in ein anderes Archiv.	DestFolder (Folder)
Delete	Verschiebt ein Dokument in den Ordner "Gelöschte Dokumente" oder löscht es endgültig.	Final (Boolean)
Move	Verschiebt ein Dokument in einen anderen Ordner oder in ein anderes Archiv.	DestFolder (Folder)
Update	Speichert Änderungen an den Eigenschaften des Dokuments.	
WriteDocument	Schreibt das Dokument auf den Datenträger.	Path (String)

So exportieren Sie ein Dokument:

```
' Pfad-Variable deklarieren
Dim sPath as String
' Schreibt das Dokument mit dem vorhandenen Dateinamen
' aus der Filename-Eigenschaft in den Ordner
' C:\Eigene Dateien\
sPath = csDoc.WriteDocument("C:\Eigene Dateien\")
' Schreibt das Dokument mit übergebenem Dateinamen
sPath = csDoc.WriteDocument("C:\Eigene Dateien\Web.html")
```

Dokumenttypen

In ContentSaver werden mehrere Dokumenttypen unterschieden. Die richtige Angabe des Dokumenttyps ist sehr wichtig, da sie die technische Art der Speicherung und der Anzeige in ContentSaver bestimmt.

ContentSaver analysiert bereits beim Speichern, um welchen Dokumenttyp es sich handelt. Kann der Dokumenttyp nicht eindeutig festgestellt werden, wird die Datei einfach mit dem Dokumenttyp "Unbekannt" gespeichert.

Konstanten für Dokumenttypen

Dokumenttyp	Konstante	Speicherung	Wert
Textdokumente	`csDocText`	Text	90
HTML-Dokumente	`csDocHTML`	Text	100
Bilder	`csDocImage`	binär	110
Stylesheets	`csDocStyleSheet`	Text	120
Skripte	`csDocScript`	Text	130
Java-Applets	`csDocCode`	binär	140
Multimedia-Dateien	`csDocMultimedia`	binär	150
Kontakte	`csDocContact`	Text	600
Ideen	`csDocIdea`	Text	500
Suchvorlagen	`csDocSearch`	Text	400
Notierte Internetadressen	`csDocLink`	Text	300
Unbekannt	`csDocUnknown`	binär	0

Hinweis: ContentSaver kann zurzeit eine Volltextsuche in folgenden Dateitypen durchführen: csDocText, csDocHTML, csDocIdea

Wichtigkeit

Jedem Dokument kann eine von vier Wichtigkeitsstufen zugewiesen werden. Standardmäßig trägt ein Dokument die Wichtigkeitsstufe „Normal“.

Konstanten für Dokument-Wichtigkeit

Wichtigkeit	Konstante	Wert
Sehr hoch	`csImportanceVeryHigh`	4
Hoch	`csImportanceHigh`	3
Normal	`csImportanceNormal`	2
Niedrig	`csImportanceLow`	1

So fügen Sie ein neues Text-Dokument hinzu:

```
sUrl = "Hans Mustermann"
sTitle = "Testdokument"
sContent = "Dies wird der Inhalt des Dokuments"
' Dokument wird hinzugefügt
Set csDoc = csArc.AddDocument(sUrl, sTitle, _
                              csDocText, sContent)
```

Die Categories-Auflistung

Alle Kategorien, die Sie in einem Archiv definiert haben (Archiv-Kategorien), können Sie über die `Archive.Categories`-Auflistung bearbeiten. Zur besseren Übersicht sind die Archiv-Kategorien wie Ordner hierarchisch in Haupt- und Unterkategorien gegliedert.

Davon zu unterscheiden sind die Kategorien, die Sie einem Dokument bereits zugewiesen haben (Dokument-Kategorien). Sie erhalten diese mit der `Categories`-Auflistung des Dokument-Objektes. Bevor Sie einem Dokument eine völlig neue Kategorie zuweisen wollen, müssen Sie diese erst zu den Archivkategorien hinzufügen.

Eigenschaften der Categories-Auflistung

Name	Beschreibung	Datentyp	R/W
`Count`	Gibt die Anzahl der Kategorien innerhalb der Auflistung zurück.	`Long`	R
`Item`	Ermöglicht den Zugriff auf eine einzelne Kategorie in der Auflistung. Der Zugriff ist auch über die `CategoryID` möglich.	`Object/ Document`	R
`Parent`	Gibt das übergeordnete Kategorie-Objekt zurück.	`Object/ Folder`	R

So greifen Sie auf Archiv-Kategorien zu:

```
' Category-Variable deklarieren
Dim csCat as CSObj.Category
' Zugriff auf alle Hauptkategorien
For Each csCat in csArc.Categories
   Debug.Print csCat.Name
   ' Die Hauptkategorie könnte Unterkategorien besitzen
   If csCat.Categories.Count Then
      ...
   End if
Next
```

So ermitteln Sie alle einem Dokument zugewiesenen Kategorien:

```
' Gewünschtes Dokument holen
Set csDoc = csArc.GetDocFromID(lEntryID)
' Zugriff auf alle Dokument-Kategorien
For Each csCat In csDoc.Categories
   Debug.Print csCat.Name
Next
```

Hinweis: Im Gegensatz zu den Archiv-Kategorien werden Dokument-Kategorien nicht hierarchisch zurückgegeben.

Methoden der Categories -Auflistung

Name	Beschreibung	Parameter
`Add`	Fügt der Auflistung eine neue Kategorie hinzu.	Name (String)

So fügen Sie eine neue Kategorie hinzu:

```
' Folder-Variable deklarieren
Dim csCat as CSObj.Category
```

```
' Neue Hauptkategorie hinzufügen
Set csCat = csArc.Categories.Add("Marketing")
' Zusätzliche Unterkategorie zu Hauptkategorie Marketing
Set csCat2 = csCat.Categories.Add("Konkurrenz")
```

Das Category-Objekt

Das Category-Objekt ist .

Eigenschaften des Documents-Objekts			
Name	**Beschreibung**	**Datentyp**	**R/W**
`Categories`	Gibt eine Auflistung aller dem Unterkategorien zurück.	`Object/ Categories`	R
`CategoryID`	Eindeutige ID der Kategorie im Archiv	`Long`	R
`Comment`	Kommentar zur Kategorie.	`String`	R/W
`Name`	Bezeichnung der Kategorie	`String`	R/W
`Parent`	Übergeordnetes Objekt der Kategorie. Bei Dokumentkategorien ist dies das zugehörige Dokument-Objekt.	`Object/ Category/ Document`	R

Methoden des Document-Objektes		
Name	**Beschreibung**	**Parameter**
`Assign`	Weist einem Dokument diese Kategorie zu.	Document (Document)
`Delete`	Löscht eine Kategorie aus dem Archiv.	
`Move`	Verschiebt eine Kategorie als Unterkategorie in eine andere Kategorie. Diese kann sich auch in einem anderen Archiv befinden.	DestFolder (Folder)
`Remove`	Entfernt eine Zuweisung der Kategorie zu einem Dokument.	Document (Document)
`Update`	Speichert Änderungen an den Eigenschaften der Kategorie.	

So weisen Sie einem Dokument eine Kategorie zu:

```
' Kategorie anhand der CategoryID auswählen
Set csCat = csArc.GetCategoryFromID(lCategoryID)
' Dokument auswählen
Set csDoc = csArc.GetDocFromID(lEntryID)
' Kategorie dem Dokument zuweisen
csCat.Assign csDoc
```

So heben Sie die Zuweisung einer Dokument-Kategorie auf:

```
csCat.Remove csDoc
```

Die Fields-Auflistung

Archive, Ordner und Dokumente können beliebig viele zusätzliche Datenfelder besitzen, die zusätzliche Informationen zum Objekt aufnehmen können. Diese Zusatzinformationen sind bisher nicht in der ContenSaver-Anwendung sichtbar. Die maximale Länge eines Feldes beträgt 255 Zeichen.

Alle Zusatzfelder eines Archivs, Ordners oder Dokuments sind über deren Fields-Eigenschaft zugreifbar.

Eigenschaften der Fields-Auflistung			
Name	**Beschreibung**	**Datentyp**	**R/W**
`Count`	Gibt die Anzahl der Kategorien innerhalb der Auflistung zurück.	`Long`	R
`Item`	Ermöglicht den Zugriff auf ein einzelnes Datenfeld in der Auflistung. Der Zugriff ist auch über den Namen möglich.	`Object/ Field`	R/W
`Parent`	Gibt das übergeordnete Objekt zurück.	`Object`	R

Methoden der Fields-Auflistung		
Name	**Beschreibung**	**Parameter**
`Add`	Fügt der Auflistung ein neues benutzerdefiniertes Feld hinzu.	Name (String)

So fügen Sie neue Zusatzfelder hinzu:

```
' Zusatzinformation "Autor" im Dokument speichern
csDoc.Fields.Item("Autor") = "Hans Mustermann"
' Zusatzinformation "bearbeitet" im Ordner speichern
csFldr.Fields.Item("bearbeitet") = "Ja"
```

So greifen Sie auf ein bestimmtes Zusatzfeld zu:

```
Dim sValue as String
sWert = csFldr.Fields.Item("bearbeitet")
sWert = csDoc.Fields.Item("Autor")
```

Das Field-Objekt

Das Field-Objekt repräsentiert ein Element der Fields-Auflistung. Die maximale Länge des Feldes beträgt 255 Zeichen.

Eigenschaften des Field-Objekts			
Name	**Beschreibung**	**Datentyp**	**R/W**
`FieldID`	Eindeutige ID des benutzerdefinierten Feldes	`Long`	R
`Name`	Name des Feldes	`String`	R
`Parent`	Gibt die Auflistung zurück, innerhalb dessen das Feld	`String`	R/W
`Value`	Inhalt des benutzerdefinierten Feldes (maximal 255 Zeichen)	`String`	R/W

Methoden des Field-Objektes		
Name	**Beschreibung**	**Parameter**
`Delete`	Löscht das benutzerdefinierte Feld des zugehörigen Objektes.	

So löschen Sie ein Zusatzfeld:

```
doc.Fields.Item("Autor").Delete

' Oder alternativ einen leeren String zuweisen
doc.Fields.Item("Autor").Value = ""
```

Programmierbeispiele

In diesem Abschnitt werden zwei sinnvolle Beispiele vorgestellt, die demonstrieren, wie die Programmierschnittstelle von ContentSaver verwendet werden kann.

Wichtiger Hinweis: Bevor Sie mit den Beispielen arbeiten oder sie für Ihre eigenen Zwecke abändern, sollten Sie immer erst eine Sicherungskopie Ihres Archivs anfertigen und die Programme mit der Archivkopie testen. Durch Programmierfehler oder Versehen könnten unbeabsichtigt Daten gelöscht oder verändert werden!

Speichern von E-Mails aus Microsoft Outlook®

Mit dem folgenden Programm können Sie in Microsoft Outlook markierte E-Mails in ContentSaver speichern. Dabei wird das aktuell markierte E-Mail-Element ermittelt und dessen Inhalt ausgelesen. Damit das Dokument später in ContentSaver bearbeitet werden kann, wird der Inhalt der E-Mail-Nachricht gegebenenfalls in das HTML-Format umgewandelt.

Das neue Dokument erhält als Adresse den Namen des Absenders und als Titel den Betreff der E-Mail. Nach dem erfolgreichen Hinzufügen des Dokuments wird eine ggf. laufende ContentSaver-Anwendung benachrichtigt, dass die Ansicht des Ordners „Neue Dokumente“ aktualisiert werden soll, damit das neue Dokument sofort sichtbar wird.

Dieses Beispiel wurde mit Outlook 2000, 2002 und 2003 getestet.

Wichtig: Vor Verwendung des Beispiels müssen in Visual Basic unter „Extras“, „Verweise“ folgende Verweise hinzugefügt werden:

- ContentSaver Objektbibliothek
- ContentSaver Shell-Komponenten
- Microsoft VBScript Regular Expressions 5.5

```
Public Sub CS_SaveSelection()
   Dim myItem As Outlook.MailItem
   Dim myInspector As Outlook.Inspector
   Dim csNS As CSObj.NameSpace
   Dim csDoc As CSObj.Document
   Dim csArc As CSObj.Archive
   Dim csFldr As CSObj.Folder
   ' ContentSaver Namespace-Objekt erzeugen
   Set csNS = New CSObj.NameSpace
   Dim lFolderID As Long
   Dim bUseDocSaver As Boolean

   Dim csSaver As CSShell.DocSaver
   Set csSaver = New CSShell.DocSaver

   ' -------------------------
   ' Objekte für Reguläre Ausdrücke erzeugen
   Dim regEx1 As VBScript_RegExp_55.RegExp
   Set regEx1 = New VBScript_RegExp_55.RegExp
   Dim match As VBScript_RegExp_55.match
   Dim matches As VBScript_RegExp_55.MatchCollection
```

```
   Dim sTemp As String, sInfo As String
   Dim sSylesheet As String
   ' Stylesheet-Vorlage, damit Plaintext-E-Mails besser
aussehen
   sSylesheet = "<STYLE TYPE='text/css'>BODY {color:
black;font-size: 14px;font-family: Verdana, Geneva, Arial,
Helvetica, sans-serif;font-style : normal;}P {margin-bottom:
.25ex; margin-top: .25ex;}</STYLE>"

   ' Ordnerauswahl-Dialogfeld anzeigen
   Set csFldr = csNS.PickFolder()
   Set csArc = csFldr.Archive
   lFolderID = csFldr.FolderID

   If lFolderID = 1 Then lFolderID = csFolderNewDocuments

   Dim Element As Object

   ' Jedes in Outlook markierte Element durchlaufen
   For Each Element In ActiveExplorer.Selection

      ' Prüfen, ob das markierte Element eine E-Mail ist
      If TypeOf Element Is MailItem Then
         Set myItem = Element
      Else
         Exit For
      End If

      ' Infos vom Kopf der E-Mail auslesen
      sInfo = "<FONT style=verdana><b>Von:</b> " &
myItem.SenderName & "<br>"
      sInfo = sInfo & "<b>To:</b> " & myItem.To & "<br>"
      sInfo = sInfo & "<b>Subject:</b> " & myItem.Subject &
"<br>"
      sInfo = sInfo & "<b>Date:</b> " & myItem.ReceivedTime &
"<br><br></FONT>"

      Set myInspector = myItem.GetInspector
      If myInspector.EditorType <> olEditorHTML Then
         ' Plaintext-Mail in HTML umwandeln
         sTemp = myItem.Body
         sTemp = Replace$(sTemp, vbCrLf, "<br>")
         sTemp = Replace$(sTemp, vbNewLine, "<br>")
         sTemp = Replace$(sTemp, "  ", "  ")
         sTemp = "<HTML><HEAD>" & sSylesheet &
"</HEAD><BODY>" & sInfo & sTemp & _
         "</BODY></HTML>"
      Else
         ' E-Mail ist bereits im HTML-Format
         sTemp = myItem.HTMLBody
         With regEx1
            .IgnoreCase = True
            .MultiLine = True
            .Global = True
            ' BODY-Tag herausfinden, um E-Mail-Adresse
            ' am Beginn des Dokuments einzufügen
            .Pattern =
"<(?:BODY)(?:""[^""]*""|'[^']*'|[^'"">])*>"
            Set matches = .Execute(sTemp)
            If matches.Count Then
```

```
               sTemp = Replace$(sTemp, matches.Item(0).Value,
matches.Item(0).Value & sInfo, , , vbTextCompare)
            End If
         End With
         ' DocSaver-Objekt zum Speichern verwenden
         bUseDocSaver = True
      End If

      Set csSaver.Archive = csFldr.Archive
      If bUseDocSaver Then
         Set csDoc = csSaver.AddDocument(csdochtml, sTemp,
myItem.SenderName, myItem.Subject, , lFolderID)
      Else
         ' Dokument über Objektmodell hinzufügen
         Set csDoc =
csFldr.Archive.AddDocument(myItem.SenderName, _
                                        csdochtml, _
                                        myItem.Subject, _
                                        sTemp, , lFolderID)
      End If

      Dim Attach As Attachment
      For Each Attach In myItem.Attachments
         sTemp = csNS.TempPath & "\" & Attach.FileName
         Attach.SaveAsFile sTemp
         Call csSaver.AddDocument(csDocunknown, "", sTemp,
Attach.FileName, , lFolderID)
      Next
   Next

   If Not csDoc Is Nothing Then
      ' notify csApp
      csArc.NotifyCSApp lFolderID
      MsgBox "Die E-Mail(s) wurde(n) gespeichert.", _
             vbInformation, "ContentSaver"
   End If

End Sub
```

Erzeugen einer CSV-Datei mit allen ContentSaver-Dokumenten

Das folgende Beispielprogramm bildet eine Komma-separierte Exportdatei mit allen Dokumenten eines Archivs. Dabei werden rekursiv alle Ordner des Archivs durchlaufen und folgende Eigenschaften aller vorhandenen Dokumente ausgegeben: ID, Ordner, Adresse, Größe, Kommentar und zugewiesene Kategorien.

Das Beispiel kann leicht individuell angepasst werden.

```
Private lFileHandle As Long
Private Const sSeparator As String = "|"
Private Const sFilePath As String = "C:\Archiv.txt"

Public Sub BuildArchiveDump()
   Dim csNS As CSObj.NameSpace
   Set csNS = New CSObj.NameSpace
   Dim arc As CSObj.Archive
   Dim sLine As String
   ' Auf das Standardsarchiv zugreifen
   Set arc = csNS.DefaultArchive
```

```
   lFileHandle = FreeFile
   ' Bereits vorhandene Datei ggf. vorher löschen
   if Len(Dir$(sFilePath)) Then Kill sFilePath
   ' Textdatei öffnen
   Open sFilePath For Append As lFileHandle
   ' Spaltenüberschrift zusammensetzen
   sLine = "DocID" & sSeparator & "Ordner" & sSeparator & _
           "Adresse" & sSeparator & "Größe" & sSeparator & _
           "Kommentar" & sSeparator & "Kategorien"

   ' Spaltenüberschrift in Exportdatei schreiben
   Print #lFileHandle, sLine
   ' rekursiv alle Ordner durchlaufen und die Dokumente
ausgeben
   Call TreatFolders(arc.Folders)
   ' Datei schließen
   Close lFileHandle
End Sub
```

Die Treatfolders-Funktion wird vielfach rekursiv aufgerufen, bis alle Ordner durchlaufen sind.

```
Public Sub TreatFolders(fldrs As CSObj.Folders, _
           Optional ByVal FolderPath As String)
   Dim fldr2 As CSObj.Folder
   Dim fldrs2 As CSObj.Folders
   Dim doc As CSObj.Document
   Dim sTempFolder As String

   ' Alle Ordner dieser Ordnerebene durchlaufen
   For Each fldr2 In fldrs
      ' Pfad speichern
      If Len(FolderPath) Then
         sTempFolder = FolderPath & "\" & fldr2.name
      Else
         sTempFolder = fldr2.name
      End If
      Debug.Print sTempFolder
      For Each doc In fldr2.Documents
         ' Eigenschaften für das einzelne Dokument ausgeben
         Call DumpDocProperties(doc, sTempFolder)
      Next
      Set fldrs2 = fldr2.Folders
      If fldrs2.Count Then
         ' Es sind Unterordner vorhanden
         Call TreatFolders(fldrs2, sTempFolder)
      End If
   Next
End Sub
```

Die Routine `DumpDocProperties` dient dazu, die einzelnen Eigenschaften eines Dokuments auszugeben. Hier könnten Sie das Dokument z.B. zusätzlich als Datei auf die Festplatte exportieren.

```
Public Sub DumpDocProperties(doc As CSObj.Document, _
                             FolderPath As String)
   Dim sLine As String
   Dim cat As CSObj.Category
   Dim sKategorien As String
   ' Kategorien ermitteln
   For Each cat In doc.Categories
      If Len(sKategorien) Then
```

```
            sKategorien = sKategorien & "," & cat.name
         Else
            sKategorien = cat.name
         End If
      Next
      With doc
         ' "DocID|Ordner|Adresse|Kommentar|Größe|Kategorien"
         sLine = .EntryID & sSeparator & FolderPath & _
                 sSeparator & .URL & sSeparator & .Size & _
                 sSeparator & .Comment & sSeparator &
sKategorien
      End With
      Print #lFileHandle, sLine
End Sub
```

Support und Kontakt

Wenn Sie irgendein Problem oder eine Frage zur Programmierung von ContentSaver haben, bitten wir Sie, ausschließlich unser Programmierforum aufzusuchen: http://www.macropool.com/forum

Die Entwickler von ContentSaver versuchen nach Kräften alle dortigen Fragen zu beantworten. Da die Wahrscheinlichkeit sehr hoch ist, dass jemand die gleiche Frage bereits einmal gestellt hat oder das gleiche Problem hatte, bitten wir Sie, zunächst die Volltextsuche des Forums zu verwenden.