Manuskript-Auszug aus:

Jörg Schmidt: "Tabellenkalkulation mit OpenOffice.org 3 - Calc"

Galileo Computing, Bonn 2009

mit freundlicher Genehmigung des Rechteinhabers

(bei Fragen: info@jm-schmidt.de)

8.1.1 Das ODF-Dateiformat nutzen

Ganz am Anfang des Buches sprach ich bereits davon, dass das von OpenOffice.org verwendete Standardformat zum Speichern von Dateien (ODF – OpenDocument Format) inzwischen ein international anerkannter ISO-Standard ist (ISO/IEC 26300). Trotz der Vorteile, die ein solcher Standard im Allgemeinen bietet, ist natürlich auch die Frage berechtigt, wie Sie das ODF praktisch nutzen können. Ein kleines Beispiel mag eine Möglichkeit der Nutzung verdeutlichen.

Beim ODF handelt es sich um ein XML-Dateiformat, welches komprimiert als Zip-Archiv vorliegt. Wenn Sie bei einer normalen *.ods-Datei die Dateiendung auf *.zip ändern, ist es möglich, das Archiv mit einem gewöhnliches Packprogramm zu entpacken, um dessen Einzelbestandteile sichtbar zu machen (siehe Abbildung 8.76).

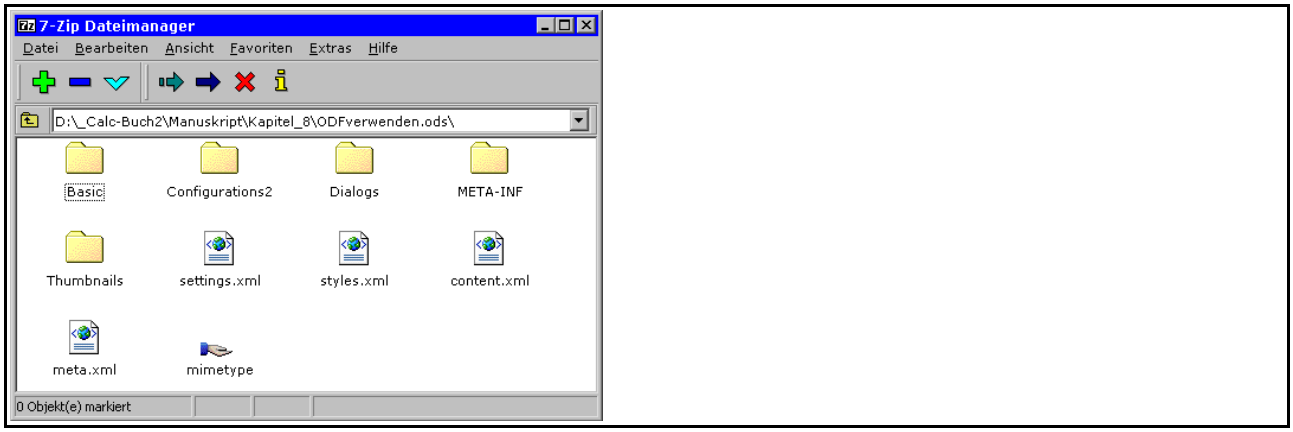

Abbildung 8.76: Inhalt eines Calc-Dokuments (*.ods)

Im Folgenden sei eine Möglichkeit aufgezeigt, wie Sie das Archiv nutzen können, um weitere Inhalte, zusammen mit einem Dokument, weiterzugeben.

Blicken Sie zurück auf das Beispielmakro zum Eintragen eines Datums (siehe Abbildung 8.71): sicherlich ein sehr einfaches Beispiel, für dessen Bedienung ein Nutzer keine Hilfe benötigen würde. Aber was wäre, wenn Sie doch eine Hilfe mitliefern wollten? Die einfachste Möglichkeit wäre es sicher, einen zweiten Dialog zu erstellen, auf welchem Sie einen (kurzen) Hilfetext anzeigen könnten. Für einfache Anwendungen mag das gehen, aber für komplexere Anwendungen ist das keine besonders befriedigende Lösung.

Es wäre nun einfach, eine Hilfedatei separat auszuliefern, aber lassen Sie uns überlegen, ob es nicht doch einen besseren Weg gibt. Da OpenOffice.org unter verschiedensten Betriebssystemen lauffähig ist, sollten Sie für Ihre Hilfedatei ein Format verwenden, welches unter den verschiedenen Systemen auch angezeigt werden kann. Für unser Beispiel nutze ich einmal das PDF. Erweitern wir den Dialog also zunächst um eine Schaltfläche, mit welcher die PDF-Datei später aufgerufen werden kann. Das Ganze könnte dann beispielsweise wie in Abbildung 8.78 gezeigt aussehen. Sie erkennen eine zusätzliche Schaltfläche zum Aufrufen der Hilfe, welche ein Hilfesymbol als Ersatz für eine Beschriftung beinhaltet.

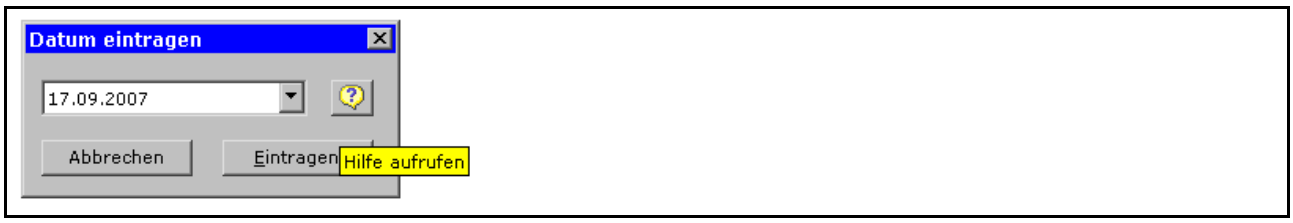

Abbildung 8.78: Erweiterter Dialog

Erstellen Sie jetzt zunächst eine Hilfedatei und speichern Sie diese als PDF ab. Mit dem in OpenOffice.org integrierten PDF-Export ist diese Aufgabe – mit Hilfe des Writers – schnell erledigt.

Schreiben Sie nun ein Makro, welches die Hilfedatei aufruft, beispielsweise:

```
Sub hilfe_aufrufen()
Dim aufrufen As Object
aufrufen = createUnoService("com.sun.star.system.SystemShellExecute")
aufrufen.execute(ConvertToURL("D:\hilfe.pdf"), "", 0)
End Sub
```

Weisen Sie dieses Makro der Hilfeschaltfläche des Dialogs zu. Es ist somit möglich, die Hilfedatei vom Dialog aus aufzurufen, wenn sie sich im angegebenen Pfad (D:\hilfe.pdf) befindet.

Um jetzt diese Datei in das bestehende Dokument zu integrieren, können Sie wie folgt vorgehen:

- Benennen Sie die Dateiendung des Dokuments in *.zip um, und entpacken Sie den Inhalt des Zips in ein Verzeichnis.
- Erzeugen Sie innerhalb des Verzeichnisses einen Ordner (nennen Sie ihn beispielsweise hilfe).
- Kopieren Sie das PDF (hilfe.pdf) in diesen Ordner. Da für die Hilfeschaltfläche ein Bild (bild.png) benutzt wird, kopieren Sie dieses Bild ebenfalls in den Ordner.
- Wechseln Sie in den Ordner META-INF (siehe Abbildung 8.76), öffnen Sie die dort enthaltene Datei manifest.xml in einem Editor, und ergänzen Sie folgende drei Zeilen:

<manifest:file-entry manifest:**media-type="meintyp"** manifest:full-path="hilfe/"/>

<manifest:file-entry manifest:media-type="" manifest:full-path="hilfe/hilfe.pdf"/>

<manifest:file-entry manifest:media-type="" manifest:full-path="hilfe/bild.png"/>

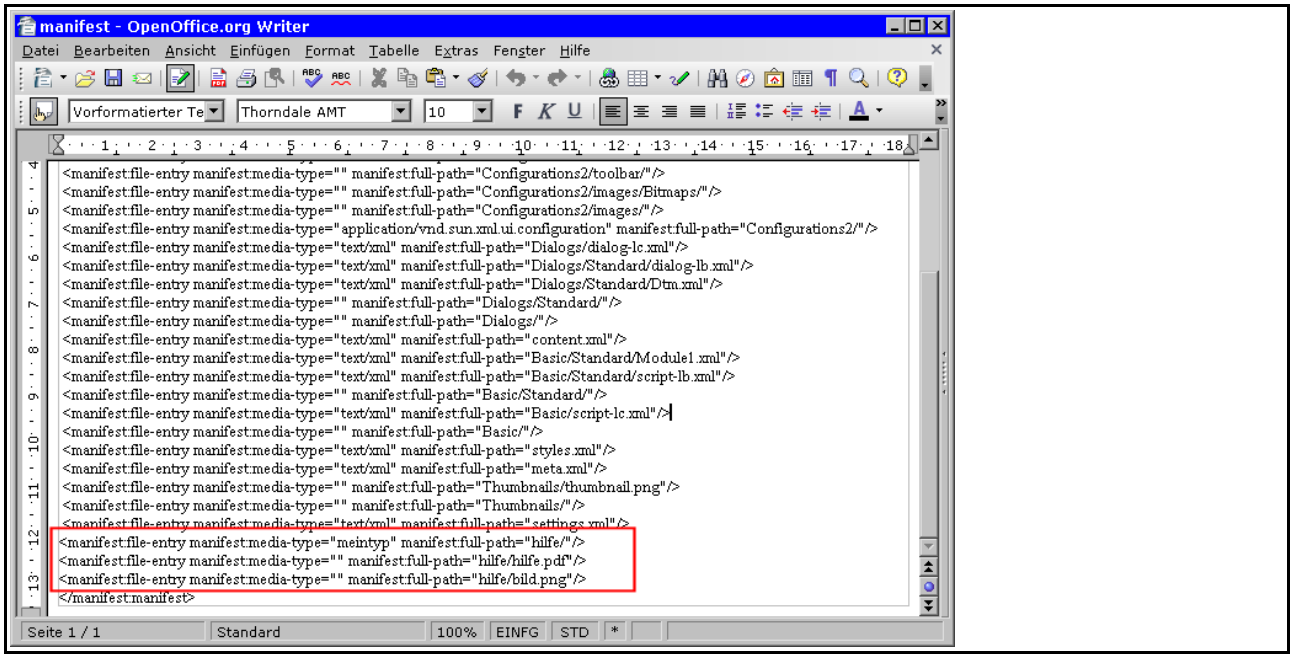

Abbildung 8.80: manifest.xml

Ich weiß nicht, welchen Editor Sie bevorzugen, notfalls können Sie die Datei manifest.xml auch mit OpenOffice.org öffnen. Folgende Schritte sind dazu notwendig:

- Wählen Sie **Datei • Öffnen**, wählen Sie die Datei manifest.xml, und klicken Sie auf **Öffnen**.
- Wählen Sie im erscheinenden Importfilter den UTF-8 Zeichensatz, und klicken Sie auf **OK**.
- Ergänzen Sie die drei oben aufgeführten Zeilen (siehe Abbildung 8.80), und speichern Sie die Änderung.

Es ist unbedingt erforderlich, dass Sie für den Ordner einen Typeintrag vornehmen, weil ihn OpenOffice.org sonst beim späteren Arbeiten mit der Datei wieder entfernen würde. Es reicht jedoch aus, wenn Sie einen ‚Phantasietyp' verwenden, ich habe beispielsweise „meintyp" benutzt.

Ihr ursprüngliches Verzeichnis sollte nun den zusätzlichen Ordner hilfe mit den zwei Dateien beinhalten (siehe Abbildung 8.82).

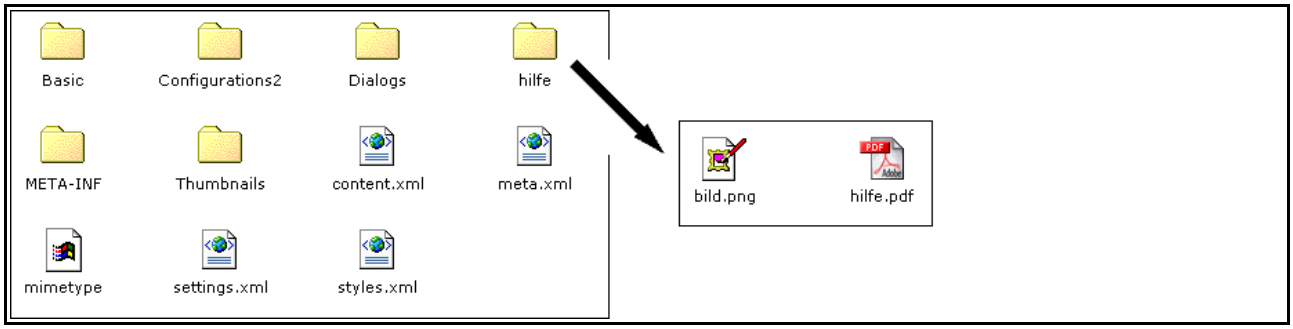

Abbildung 8.82: Hinzugefügter Ordner

Packen Sie dieses Verzeichnis wieder als Zip ein, und benennen Sie das Zip anschließend in die ursprüngliche Dateiendung (z.B. *.ods) um.

Sie verfügen nun über die ursprüngliche Datei, die sich in OpenOffice.org normal öffnen lassen sollte. Wechseln Sie in die Basic-IDE, und ergänzen Sie den Code der Dokumentbibliothek:

```
Dim dialog1 As Object


Sub initialisieren()
```

```
Dim args(0)
tmp = ermittle_pfad()
z = _
createUnoService("com.sun.star.packages.Package")
args(0) = ThisComponent.URL
z.initialize(Args())
ebene = z.getByHierarchicalName("hilfe")
alles = ebene.getElementNames()
schreiben = _
createUnoService("com.sun.star.ucb.SimpleFileAccess")
For i = LBOUND(alles()) To UBOUND(alles())
 stream = _
 z.getByHierarchicalName("hilfe/" & alles(i)). _
  GetInputStream()
 schreiben.WriteFile(tmp & "/" & alles(i), stream)
Next i
End Sub

Sub dlg_start()
tmp = ermittle_pfad()
DialogLibraries.LoadLibrary("Standard")
dialog1 = CreateUNODialog(DialogLibraries.Standard.Dtm)
dialog1.GetControl("Datum").Date = cDateToIso(Date)
dialog1.Model.GetByName("cmd_hilfe").ImageURL = _
 tmp & "/bild.png"
dialog1.Execute()
End Sub

Sub datum_eintragen()
zelle = ThisComponent.GetCurrentSelection()
If zelle.supportsService("com.sun.star.sheet.SheetCell") Then
 aktuell = dialog1.GetControl("Datum").Date
 aktuell = cDateFromIso(aktuell)
 zelle.FormulaLocal = aktuell
End If
dialog1.EndExecute()
End Sub

Sub hilfe_aufrufen()
tmp = ermittle_pfad()
Dim aufrufen As Object
aufrufen = _
createUnoService("com.sun.star.system.SystemShellExecute")
aufrufen.execute(tmp & "/hilfe.pdf", "", 0)
End Sub
```

```
Sub entfernen()
tmp = ermittle_pfad()
If FileExists(tmp & "/hilfe.pdf") Then
 Kill tmp & "/hilfe.pdf"
End If
If FileExists(tmp & "/bild.png") Then
 Kill tmp & "/bild.png"
End If
End Sub

Function ermittle_pfad()
pfad = _
createUnoService("com.sun.star.util.PathSettings")
ermittle_pfad = pfad.temp
End Function
```

Rufen Sie **Extras • Anpassen...** auf, und weisen Sie das Makro initialisieren() dem Ereignis Dokument öffnen zu. Weisen Sie das Makro entfernen() dem Ereignis Dokument schließen zu. Speichern Sie abschließend die Datei.

Im Ergebnis verfügen Sie nun über eine Datei, welche über ein integriertes PDF zur Hilfedarstellung verfügt. Beim Öffnen der Datei werden automatisch das Hilfe-PDF und das Bild für die Hilfe-Schaltfläche aus dem Dokument-Archiv ins aktuelle Temp-Verzeichnis entpackt. Beide Dateien stehen somit dem Makro zur Verfügung. Beim Schließen der Datei werden die beiden entpackten Dateien wieder gelöscht.

Sie können sich sicher weitere Anwendungsfälle denken, für die das gerade beschriebene Verfahren nützlich sein kann. Denn natürlich können Sie auch beliebige andere Dateien in geschilderter Weise in das Archiv integrieren, welche dann beim Öffnen der Datei zur Verfügung stehen.

Die komplette Beispieldatei finden Sie auf der Buch-CD unter dem Namen ODFverwenden.ods.