Warum ich die lokalen Variablen **l_Horz** und **l_Vert** nicht direkt berechnete, sondern sie erst aus der Mausposition bestimmen lasse? Weil dieser **mouseover**-Befehl ein bisschen heikel ist und ich ihn deshalb lieber möglichst in Frieden arbeiten lasse, jeder **BREAKPOINT** am falschen Platz verscheucht diese Angaben, wie auch jede kleine Berührung auch die Mausposition verändern kann. Es ist also sehr einleuchtend, dass die Maus-Angaben höchst flüchtig sind, nicht wahr?

Die Addition mit der Fensterecke **$cwind.$top** führe ich deshalb durch, weil diese Variablen vom **mouseover**-Befehl relativ zum obersten Fenster angegeben werden, nicht in absoluten Angaben. Im anderen Fall akzeptiert der **$open**-Befehl zwar diese Werte auch problemlos, doch kann er nicht wirklich was damit anfangen und öffnet das Fenster immer an seinem Standard-Platz. Das wiederum ist genau am Ort des normalen Navigators und stapeln wollte ich die Navigatorbilder sicher nicht!

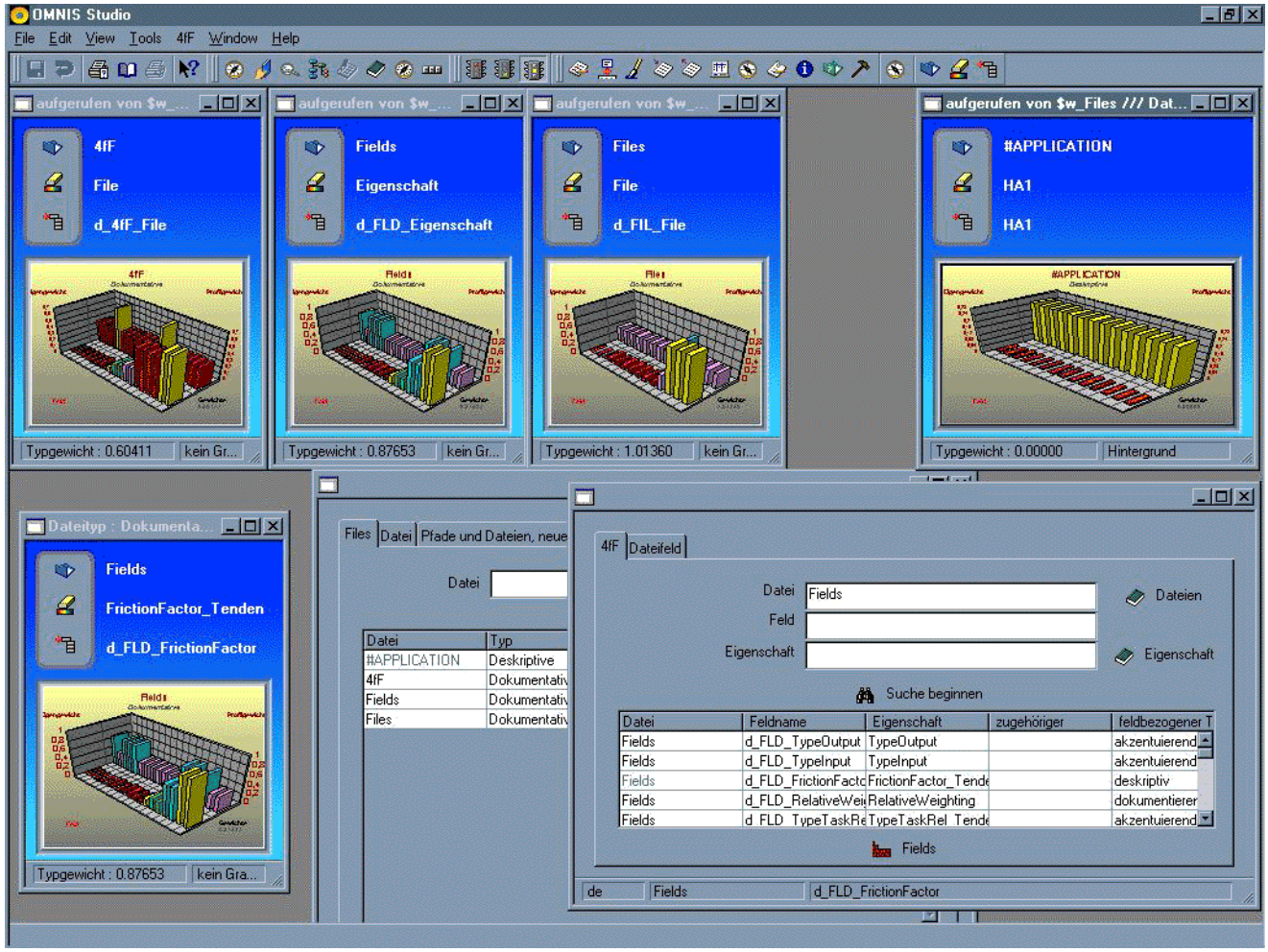

Abbildung 348 w_Files-Fenster in Aktion mit Steuerkonsole und Sonderanfertigungen

So - jetzt kann ich endlich an meine „richtigen“ Dateien gehen!

6.4 Die echten Echtdaten

Früher mal hatte ich ein Datengerüst für eine kleine Projektverwaltung vollständig durchgerechnet, an der ich nun meine neue Applikation ausprobieren will. Durch die bestehenden Vergleiche rechne ich mir gute Chancen

aus, noch vorhandene Fehler aufzufinden.

Das Erste, was ich jedoch wirklich tue - ich füge in meiner Erstellungsroutine für neue Dateien in das kleine Eingabefenster **Prompt for input** die Cancel-Option als Abbruch-Möglichkeit ein, weil ich bei der ersten Datei natürlich vergessen hatte, den Ablauf zu kontrollieren. Das wiederum wollte ich tun, weil ich mir ausrechne, den Tentakellisten-Fehler auf diesem Weg in den Griff zu kriegen.

Leider waren mir nun Datensätze bereits erstellt worden und mein Ausgangszustand demnach nicht mehr derselbe. Da ich den ganzen Prozess wieder klinisch rein von vorne beginnen wollte, musste ich auf eine Datensicherung zurückgreifen, doch aus obskuren Gründen klappen meine ganzen Suchen dann nicht mehr. Warum ich das nicht ändere? Weil es sofort wieder klappt, wenn ich per Debug das Problem suche. Ich schätze also, dass es ein interner Steuerungswert von Omnis ist, ich habe jedoch keine Ahnung, welcher. Das Problem der „stolpernden Anbindung“ an Datensicherungen steht schon länger in der **duties&beauties**, ist aber kein wirkliches Hindernis für meine Anwender, die aus dem Kreis der Datenbank-Entwickler stammen.

Zurück zu meinem **w_Files**-Fenster. Die neue Abbruch-Möglichkeit ist nichts weiter als ein Parameter in dem **Prompt for input**-Befehl, den ich natürlich abfragen muss in dieser internen **w_Files**-Methode **NewFile**:

w_Files, NewFile

```
Prompt for input [l_String]/[l_Title] Returns l_Instances (Cancel button,Prompt above entry)
    If flag false
        Quit method
    End If
```

Danach kann ich mich den Fehlern zuwenden, die mir bei der Erstellung der letzten neuen Datei aufgefallen sind. Zwei NULL-Werte sind mir bei der #APPLICATION vorgekommen, die Partner und die Tentakellisten. Den Fehler des Partnerfeldes halte ich vorläufig für kein Problem, solange ich nötige Änderungen erfolgreich durchführen kann, doch die Tentakellisten benötigen einen genauen Blick auf die individuellen **Fields**-Methoden, dort habe ich die **$NewRecord** in Verdacht, nicht sauber zu arbeiten.

Aber alle Befehle klappen wunderbar mit der HASH-Variablen **#L1**. Ich benutze sie testweise als Zwischenlager, weil sie sich bequemer ansehen lässt als ROW-Listen oder gar Binärfelder. Ein Fehler kann damit nur an der benutzten Variable **p_Row_Field.6** liegen. Also füge ich eine temporäre Liste dazwischen, die die Berechnung erst einmal aufnimmt und danach erst in das Feld von **p_Row_Field.6** einstellt, das scheint zu klappen. Der Ort dieser Änderung ist der Verarbeitungszweig **Case** 1 der **$NewRecord** der individuellen **Fields**-Methoden

Wohlgemerkt - ich bin noch in voller Aktion! Ich suche derweil durch meine SAVE- und die Arbeits-Library, ich sehe mir meine SCHEMA-Definition an, bloß um erstaunt festzustellen, dass dieses blöde Feld sehr wohl als Liste definiert ist, ich steppe vor und zurück, füge Variable ein und entferne sie wieder - und mache nach geglückter Reparatur in meinem Programmablauf gerade weiter, als wäre nichts geschehen. Das finde ich einfach gut, aber sicher kennen Sie das auch von Ihrer „Jedermanns Lieblingskind“. Doch es gibt immer noch andere Entwicklungsumgebungen, in denen jede Programmänderung erst einmal aus dem Source-Code in den Objektcode geschafft werde muss, für jeden Versuch immer wieder und wieder Testläufe erforderlich sind und im Gedenken an solche Arbeitsbedingungen freue ich mich eben noch jedesmal, Studio benutzen zu können.

Endlich habe ich meine erste Datei in meiner kleinen Applikation. Bisher habe ich keine einzige individuelle Bewertung durchgeführt und doch ist sie schon typisiert als „Dokumentative“. Das ist sehr ungewöhnlich und auch nur aus einer ungewöhnlichen Konstruktion zu verstehen. Sie ist nämlich die Mitarbeiter-Datei der kleinen Projektverwaltung, die auf nur zwei Personen ausgelegt war. Allein diese geringe Anzahl von Datensätzen macht

diese Datei zur Dokumentative, weil in einer Datei mit zwei Sätzen das Wiederfinden eines einzelnen Datensatzes absolut problemfrei ist.

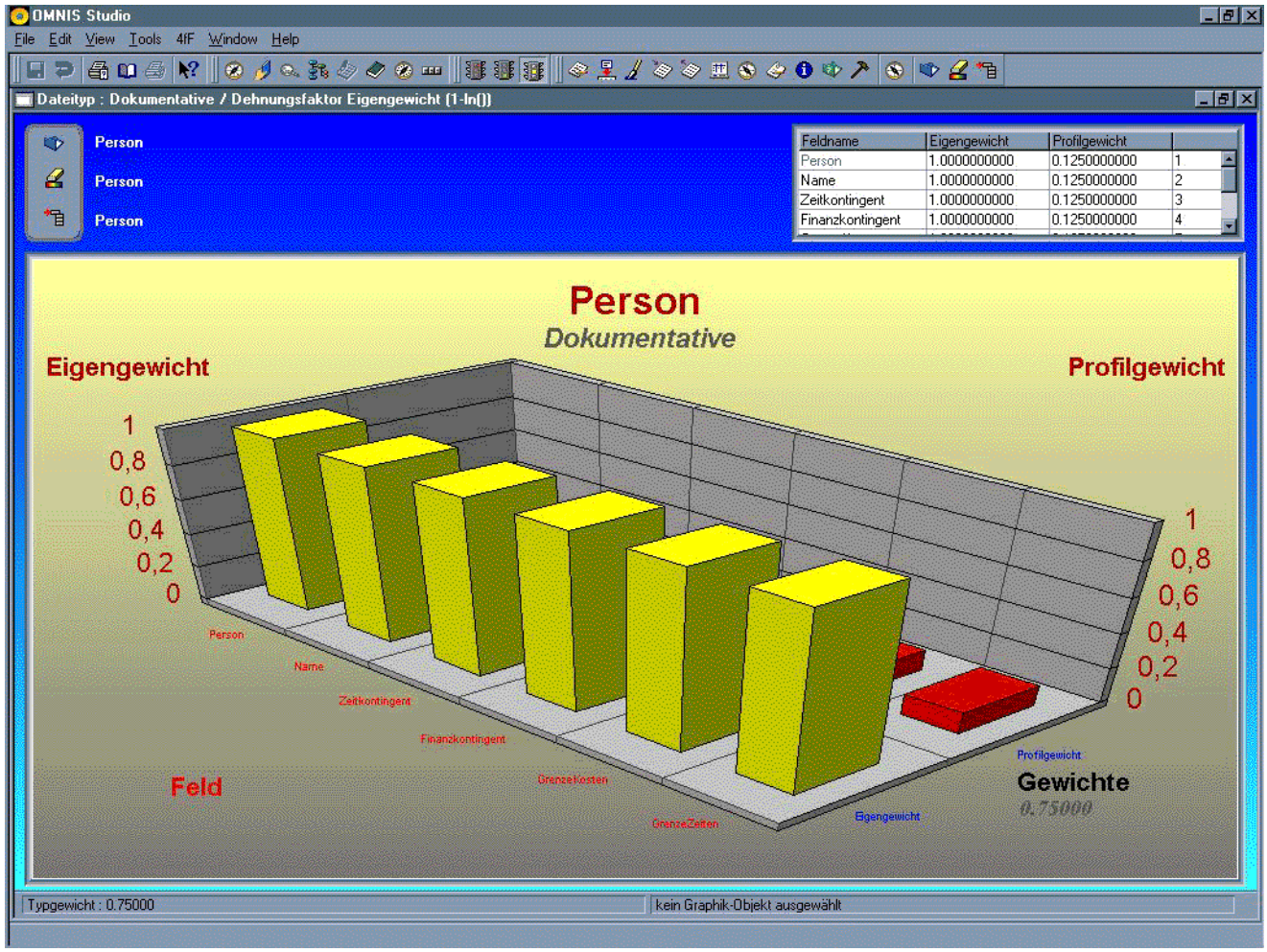

Abbildung 349 Neu-Erstellung einer Datei mit einer sehr geringen Zahl an Datensätzen (untypisch)

Diese niedrige Anzahl limitiert natürlich auch die möglichen Werte und so muss ich außer der Eigenschaft für den Namen vorerst gar nichts ändern. Bei dieser Änderung stoße ich dann auf eine weitere Korrektur, die meine individuelle **Fields**-Methode **$NewRecord** benötigt. Ich erstelle dort nämlich meine Tentakelliste aus der aktuellen Verknüpfungsliste, doch die ist eben gelegentlich nicht aktuell. Andererseits habe ich doch genau dasselbe in Grün bereits vorliegen! Die Liste **l_LST** ermittelt sich alle passenden **4fF**-Datensätze und ist damit so aktuell wie nur möglich.

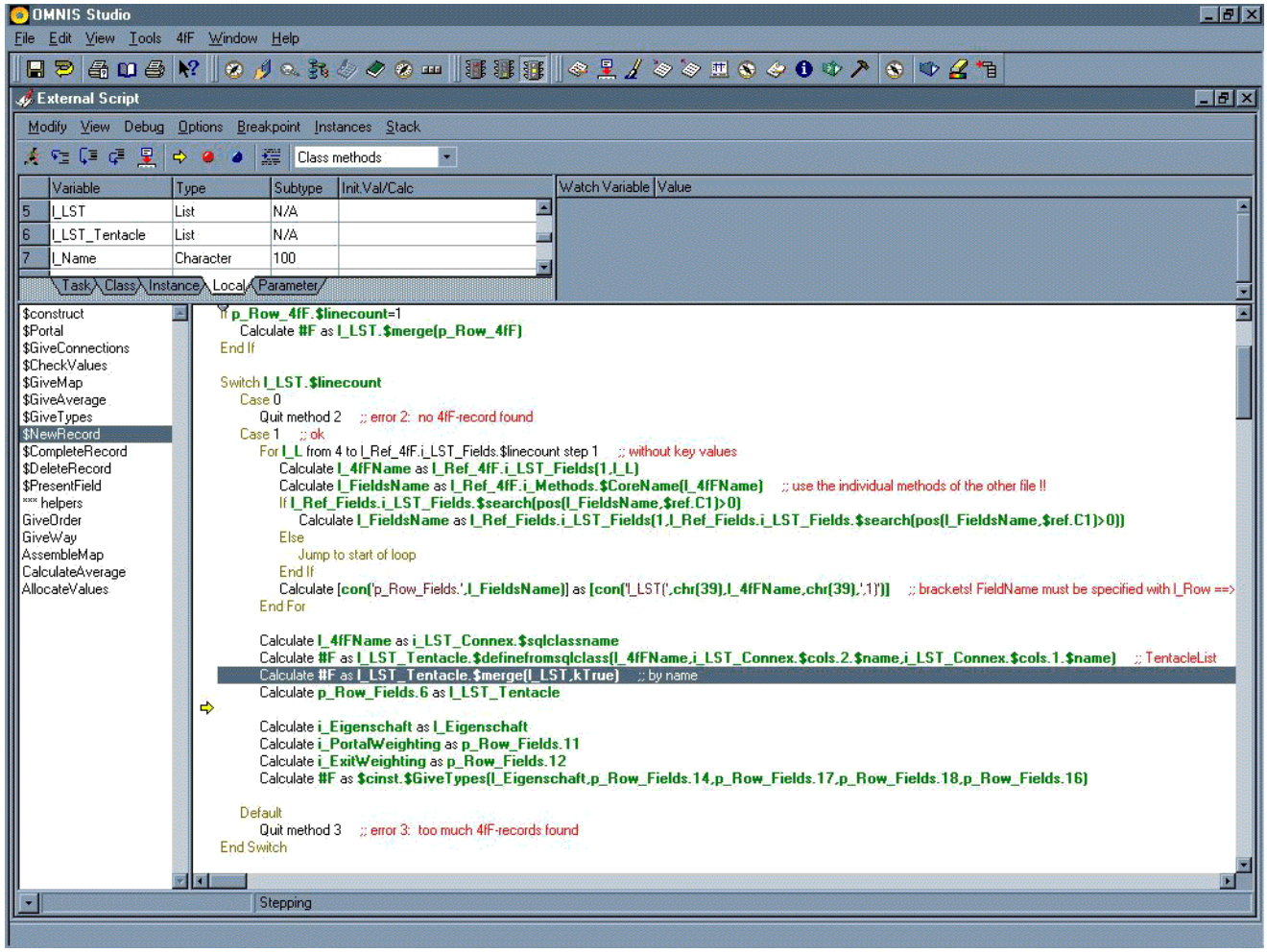

Abbildung 350 Geänderte Tentakellistenbestimmung, individuelle Fields-Methode $NewRecord

Noch 'ne Kleinigkeit. Im **w_Files**-Fenster wird meine neue Datei nicht ordentlich für die **TREELIST** der Datenpfade aufbereitet. Als erstes fällt mir auf, dass die Ermittlung des Pfades in der **$InitPage2** zu spät geschieht, ich verpflanze sie deshalb direkt hinter die Bestimmung von **i_Partner** an den Anfang der Methode:

w_Files, $InitPage2

```
If i_LST_MyList(9,i_L)<>"
    Calculate i_Path as i_LST_MyList(9,i_L)
Else
    Calculate i_Path as $itasks.//4FF//.i_SessionOmnis
End If
```

Das war's auch schon.

Nun gehe ich an die nächste Datei, dort muss ich echte Änderungen durchführen. Und schon bei der ersten hapert's. Ich habe meine beiden neuen Dateien nämlich gleichzeitig in der Übersicht des **w_4fF**-Fensters und möchte dieselbe Eigenschaft für je ein Feld beider Dateien verwenden. Meine Prüfung in der internen Methode **CheckRowField** des **w_4fF**-Fensters jedoch ist hierfür zu eng und verbietet es mir rigoros. Der **$filter**-Befehl war

nicht exakt genug verfaßt, also ergänze ich ihn, sodass er nur noch Felder derselben Datei überprüft:

w_4fF, CheckRowField

```
Calculate l_Field as i_LST_MyList(2,i_L)

Calculate l_L as i_LST_MyList.$filter(($ref.C3=i_Eigenschaft)&($ref.C2<>l_Field))
```

Noch ein zweites Problem habe ich. Ist die Änderung der Eigenschaft akzeptiert und die Anzahl der Tentakel entsprechend erhöht, werden alle nachfolgenden Felder mit derselben Erhöhung beglückt, obwohl sie überhaupt nichts damit zu tun haben. Warum? Die höhere Zahl steht ärgerlicherweise auch in der Datenliste **i_LST_MyList**, ist leider nicht nur ein Rechenfehler in meinem Fenster. Den Fehler lokalisiere ich in der individuellen **4fF**-Methode **$DoEigenschaft**. Hätte mir eigentlich früher schon auffallen müssen! Dass es das nicht getan hat, lag daran, dass es nicht bereits beim ersten Aufruf falsch lief, sondern erst beim nächsten. Es ist der **$search**-Befehl, der mir nun zeigt, dass in der Tentakel-Liste des Kunden absoluter Matsch steht.

Doch vorher fällt mir schmerzlich auf, dass die Eigenschafts-Übersicht meine neuen Eingaben gar nicht kennt. Sie findet sie sofort und fehlerfrei, wenn ich gezielt nach einem Satz greife, doch bei einer allgemeinen Suche sehe ich nur die alten Datensätze. Warum? Weil ich die Datenbank nur nach Chronologie absuche, wenn ich keinen speziellen Wert verlange. Das heißt, dass ich die Datensätze von meinen SQL-Suchaktionen in der Reihenfolge erhalte, in der sie in die Datei geschrieben wurde. Nun, es gibt bessere Sortierungen, nicht wahr? Beispielsweise nach Namen. Und das will ich nun auch tun. Das ist aber weiter kein Problem, denn das bedeutet nur eine Ergänzung der internen Datei-Task-Methode **CreateSQL**. Sie erstellt den SQL-Befehl, also muss sie auch sagen, nach welchem Feld sortiert werden soll. Ich will es nicht zu aufwändig treiben, also sortiere ich nur nach dem ersten Feld. Für diese Sortierung verwende ich eine eigene Text-Variable, die direkt nach dem **$filter**-Befehl für **i_LST_Fields**, der sich auf die Schlüsselfelder konzentriert, eingelesen wird:

Startup-Task, CreateSQL

```
Calculate #F as i_LST_Fields.$filter($ref.C3=kTrue)

Calculate l_Order as con(' order by ',i_LST_Fields(1,1))
```

Dann muss ich diesen Befehl nur an die Selektion drankleben und kriege den gewünschten Befehl:

```
Quit method con(l_Command,l_Order)
```

Nur leider gefällt mir gar nicht, was ich dann sehe! Klar, die Liste der gefundenen Eigenschaften ist nun viel ansehnlicher, doch die Tentakelliste hat es total zerhagelt. Was war denn so falsch an meinen Änderungen?

Vielleicht gar nicht? Meine Änderungen sind nicht mehr da, ich hatte wohl vergessen, sie in den Methodenfeldern auch zu speichern und irgendwann die Applikation geschlossen. Das ist nun mal das Gute und das Schlechte an Instanzen, sie existieren nur temporär.

Was tue ich also? Ich lösche die Sätze meiner ersten, kleinen Datei, denn den ganzen Neu-Anfang muss ich nun nicht mehr überprüfen, nur die Erstellung der Tentakellisten erscheint noch problematisch. Dann korrigiere ich meine **$NewRecord**s der individuellen **Fields**-Methoden ein zweites Mal und starte den ganzen Vorgang neu. Wieder sieht alles wunderbar aus und diesmal kann ich auch die Funktionalität „Änderung von Eigenschaften“ bis auf die Knochen prüfen. Bequemlichkeitshalber hatte ich nicht die Eigenschafts-Datei gelöscht und deshalb wurden die Eigenschaften für die „neuen“ Sätze eindeutig. Sie erinnern sich? Bei ganz neuen Dateien wird die Eigenschaft aus den Feldnamen hergeleitet. Kommt dieser Wert bereits einmal vor, wird aus „Eigenschaft“ dann „Eigenschaft_Dateiname“.

Nun fällt mir jedoch noch eine Kleinigkeit auf. Bei den Eigenschaften habe ich inzwischen weit mehr als 50 Datensätze und bräuchte die Möglichkeit die weiteren Sätze einzulesen - doch das ist einfach noch nicht im Fenster **w_Fields** integriert. Ich habe das bei aller Kopiererei total vergessen - das ist ein bisschen der Nachteil, wenn

man sonst Kernel-Technologie benutzt: man geht zu selbstverständlich davon aus, dass all diese generellen Funktionalitäten vorhanden sind.