Ansichtsgestaltung mit LotusScript

Neben der bekannten Formelsprache stehen uns im Domino / Notes Umfeld noch einige weitere Möglichkeiten der Anwendungsentwicklung zur Verfügung. Neben der noch relativ jungen Unterstützung von Java und XML finden wir bereits seit der Version 5 ebenfalls als eine Art Säule Lotus Script in Domino implementiert.

Lotus Script hat sich dabei in den letzten Versionen von Domino und Notes in den Vordergrund geschoben und bildet heute die Grundlage für eine Vielzahl von Anwendungen, welche auf Basis von Lotus Script entwickelt worden sind.

IBM ist auch in diesem Bereich bemüht, die Entwicklung fortzuführen. Während in der Version 7 wesentliche Erweiterungen in Lotus Script beispielsweise in Bezug auf die XML Verarbeitung realisiert wurden, so finden wir beispielsweise ab der Version 6 eine interessante Möglichkeit, Lotus Script für die Erstellung von dynamischen Ansichten einsetzen zu können.

Neben den Ihnen sicher bekannten Möglichkeiten der Erstellung von Ansichten mit den Werkzeugen des Domino Designers bietet IBM seit der Version 6 alternativ zu Möglichkeit, Ansichten auch dynamisch mit Hilfe von Lotus Script zu erzeugen. „Dynamisch" kann sogar bedeuten, dass man Anwendern mit Hilfe einer Vorgabemaske die Möglichkeit bietet, die einzelnen darzustellenden Spalten auszuwählen und diese mit Parametern (z.B. Spaltenposition, Darstellungsformen) individuell zu selektieren.

Doch wozu das Erstellen von Ansichten mit LotusScript?
Wie verfahren Sie heute, wenn Sie in einer Anwendung neue Informationen aus einer Datenbank darstellen wollen? Genau: Sie erstellen Masken und Ansichten. Aber hat der Entwickler und Anwender neben den Möglichkeiten des Verschiebens und bedingten Hinzufügens von Spalten in die Ansicht wirklich die Möglichkeit, bei Bedarf den Inhalt der Ansicht anpassen zu können? Leider lautet die Antwort wohl nein.

Wir kennen ihn aus vielen Unternehmen: Die schnelle Anforderung „mal eben" neue Informationen zugänglich zu machen. Solche Anforderungen können zum Teil mit Hilfe der dynamischen Ansichten vereinfacht werden. Diese lassen sich dann erfahrenden Notes Benutzern – und natürlich auch Domino Administratoren bzw. Domino Anwendungsentwickler – zur Verfügung stellen

Bevor wir uns allerdings an die Erstellung von dynamischen Ansichten begeben, wollen wir zunächst die Voraussetzungen definieren, welche ich hier als „gegeben" zu Grunde lege.

Grundlage für die Verwendung der dynamischen Ansichten sind die notwendigen Berechtigungen, welche es dem jeweiligen Benutzer ermöglichen, die Anpassungen an der Datenbank durchführen zu können. Dabei benötigen wir auch die Zugriffsrechte für das Erstellen von gemeinsamen Ordnern und Ansichten innerhalb der Datenbank!

Weiterhin basiert die „Dynamisierung" auf Vorgaben, welche in Form einer Maske bereitgestellt werden. Mit diesen Vorgaben können wir steuern, welche Inhalte Benutzer für deren Ansicht verwenden können.

Was der jeweilige Benutzer nicht benötigt, sind tief greifende Kenntnisse in dem Aufbau von Ansichten oder gar der Programmierung im Domino Umfeld.

Seit der Version 6 steht in dem Domino Designer eine neue Methode *„CreateView"* zur Verfügung. Diese ist ein Bestandteil der Klasse *„NotesDatabase"* und dient, wie es der Name bereits erahnen lässt, dem Erstellen von Ansichten. Als „neue Ansicht" kann dabei auch eine bestehende Ansicht als Basis für einen Kopiervorgang verwendet werden.

Mit der Methode „CreateView" stehen uns vier verschiedene Parameter zur Verfügung, deren Bedeutung ich Ihnen nachfolgende aufzeigen möchte:

- ViewName
  Dieser Parameter dient der Angabe des Namens für die neu zu erstellende Ansicht. Wenn Sie diesen Parameter nicht füllen, vergibt Notes einen Standardnamen „untitled“.

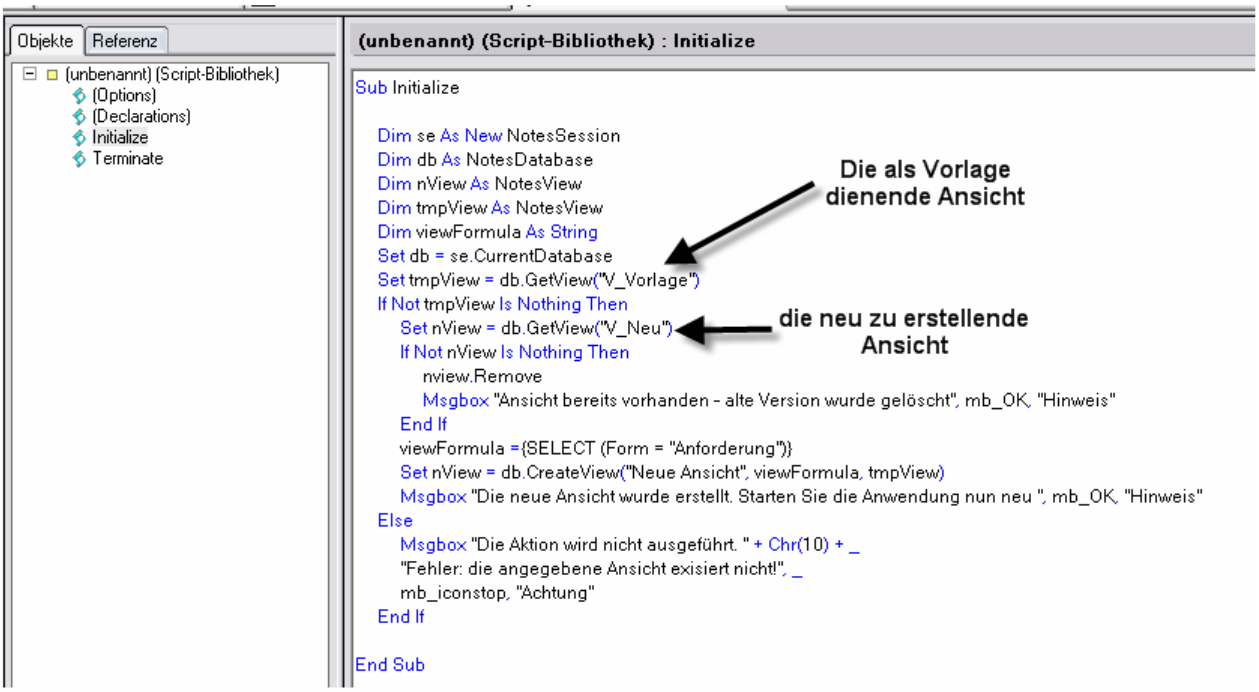
- ViewSelectionFormula
  Mit diesem Parameter legen wir die Ansichtsauswahlformel fest. Als Standard wird die Anzeige aller Dokumente durch die Formel Select @All genutzt. Alternativ dazu können Sie individuelle Vorgaben tätigen – diese erfolgen in der Formelsprache von Notes.

- TemplateView
  Wenn die neue Ansicht nicht von Grund auf neu generiert werden soll, dann können wir auch als Basis eine bereits bestehende Ansicht verwenden. In diesem Parameter „Template View“ lässt sich die Kopiermutter in Form des Ansichtsobjektes angeben. Basierend auf dieser Ansicht wird anschließend die neu zu generierende Ansicht erstellt.

  Wie auch bei den übrigen Parametern verwendet Notes einen Default-Wert, der dann zum Einsatz kommt, wenn Sie diesen Parameter nicht füllen sollten. In diesem Fall wird als Grundlage verwendet. Ist diese nicht festgelegt, werden die Grundeinstellungen der Datenbank für das Erstellen der Ansicht verwendet.

- ProhibitDesignRefresh-Modifications
  Dieser Parameter, der als boolescher Parameter anzugeben ist, legt fest, ob das Design der Ansicht bei einer Gestaltungsaktualisierung des Designer Task aktualisiert wird. Der Unterlassungswert ist mit „True“ voreingestellt (damit werden die Gestaltungselemente dieser Ansicht nicht durch automatische Gestaltungsänderungen überschrieben.)

In dem folgenden Beispiel möchte ich Ihnen nun zeigen, wie wir eine dynamische Ansicht erstellen. Dabei verwenden wir eine bereits bestehende Ansicht „V_Vorlage“ als Grundlage für die Erstellung der neuen Ansicht.



001 Das Script legt eine neue dynamische Ansicht an

In dem Script deklarieren wir zunächst wie gewohnt die Objektvariablen, welche wir später bei dem Parameter „TemplateView“ benötigen. Die Zuweisung der Variablen führen wir mit diesem Codefragment durch:

```
Set tmpView = db.GetView("V_Vorlage")
```

Es ist zu empfehlen, den Parameter „TemplateView“ zu verwenden – damit werden zusätzliche Arbeitsschritte vereinfacht.

Sicherheitshalber prüfen wir, ob die als Grundlage gewählte Ansicht auch tatsächlich existiert. Wird die Ansicht wider Erwarten nicht gefunden, geben wir eine entsprechende Fehlermeldung aus. Anschließend erfolgt eine Prüfung, ob die neu zu erstellende Ansicht bereits existiert. Sollte dies der Fall sein, dann muss die bereits bestehende Ansicht zunächst gelöscht werden.

```
Set nView = db.GetView("V_Neu")
                    If Not nView Is Nothing Then
                            nview.Remove
                            Msgbox "Ansicht bereits vorhanden - alte Version wurde gelöscht", mb_OK, "Hinweis"
                    End If
```

Jetzt wird die Ansichtsauswahlformel definiert und der Variablen *viewFormula* zugewiesen.

```
viewFormula ={SELECT (Form = "Anforderungl")}
```

Nun fehlt nur noch das Wesentliche – nämlich das Erstellen der neuen Ansicht. Dieser ordnen wir den Namen *V_Neu* zu Nachdem die gewünschten Parameter definiert sind, können Sie die Ansicht unter Verwendung der Methode *CreateView* des Datenbankobjekts und der Parameter erstellen.

Jetzt speichern Sie diese Angaben und starten die Datenbank neu – danach sind diese Änderungen aktiv.

Damit haben wir eine 1:1 Kopie einer bereits vorhandenen Ansicht angelegt. Diese wollen wir nun in einem nächsten Arbeitsschritt modifizieren und der neuen Ansicht individuelle Elemente zuordnen.

Das Anpassen einer Ansicht bezieht sich häufig auf das Anpassen der Spalten. Dazu steht uns erneut die Klasse „NotesView“ zur Verfügung.

Mit der Klasse „NotesViewColumn“ lassen sich die einzelnen Spalteneigenschaften definieren.
Unter anderem können wir in diesem Zusammenhang die Spalteneigenschaften mit den folgenden Attributen beeinflussen:

’

- *FontColor*
- *FontFace*
- *HeaderFontColor*
- *HeaderFontFace*
- *HeaderFontStyle*

Neben diesen Attributen stehen ebenfalls einige neue Methoden zur Verarbeitung der Spalten zur Verfügung:

- CreateColumn
  Neue Spalten lassen sich mit der Methode *CreateColumn* erstellen.

  Der optionale Parameter „Pos“ definiert die Position der neuen Spalte, bezogen auf eine angegebene Spalte. Verzichten Sie auf die Angabe dieses Parameters, dann wird die neue Spalte rechts in der Ansicht an die bereits bestehenden Spalten angefügt.

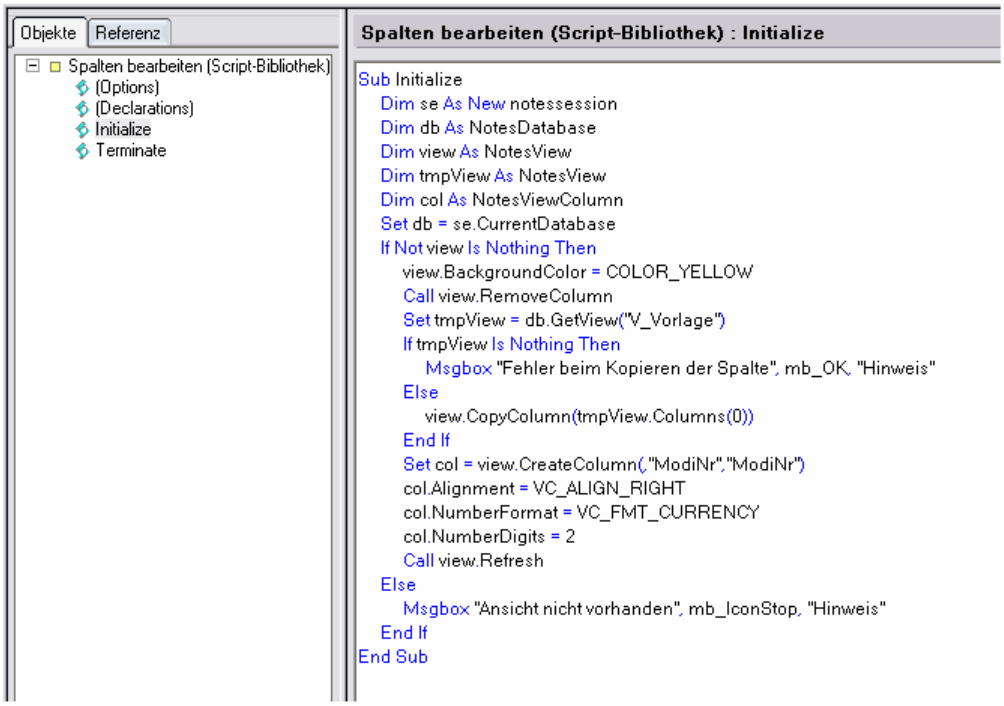
  Der Parameter „Name“ definiert die Spaltenüberschrift.

Optional können Sie mit dem Parameter „Formula“ Spaltenformeln zuweisen. Sollten Sie auf die Angabe einer eigenen Formel verzichten wollen, dann setzt Notes automatisch den Unterlassungswert „@DocNumber“ ein.

- CopyColumn
  Diese Methode dient dem Kopieren bereits bestehenden Spalten in die Ansicht. Der dabei zu verwendende Parameter „Source“ definiert die zu kopierende Spalte. Mit dieser Methode ist es möglich Spalten aus derselben, aber auch aus anderen Ansichten zu implementieren

  Den Parameter „Pos“ haben wir bereits zuvor kennen gelernt. Er dient auch hier zur Bestimmung der Position der Spalte.

- RemoveColumn
  In einer Ansicht enthaltene Spalten können mit der Methode „RemoveColumn“ entfernt werden Mit dem optionalen Parameter *Column* bestimmen Sie die zu löschende Spalte. Dabei können Sie wahlweise als Parameter die Position oder den Spaltennamen verwenden. Wenn Sie keine Angabe in Bezug auf die zu löschende Spalte vornehmen, dann wird mit dieser Methode die letzte Spalte aus der Ansicht entfernt.



002 Script zum Manipulieren der Spalten

In diesem Beispielscript verarbeiten wir die zuvor erstelle Ansicht. Diese zu bearbeitende Ansicht wird mit Hilfe der Variablen „view“ zugewiesen. Natürlich prüfen wir auch hier, ob die angegebene Ansicht vorhanden ist und geben notfalls auch eine Fehlermeldung aus.

In diesem Script passen wir nicht nur die Spalten, sondern auch die gesamte Darstellung der Ansicht an, indem wir die Farbdefinitionen ändern. Dazu verwenden wir die Eigenschaft „*BackgroundColor”*.

view.BackgroundColor = COLOR_YELLOW

Mit der Anweisung „RemoveColumn“ löschen wir eine Spalte aus der Ansicht. Da wir dieser Anweisung keinen Parameter mitgeben ,wird somit automatisch die letzte Spalte entfernt:

```
Call view.RemoveColumn
```

Schließlich fügen wir der Ansicht noch eine neue Spalte ein:

```
view.CopyColumn(tmpView.Columns(0))
```

Nachdem Sie Ihre Änderungen gespeichert haben, können wir die neue Funktion nutzen. Damit haben wir die Grundlage geschaffen, Ansichten flexibel zu gestalten.