

LazReport Reports in einer Datenbank speichern

Es kann von Vorteil sein Reports mittels einer Datenbank zu verwalten bzw. in einer Datenbank zu speichern. Einer davon ist dass es in Mehrplatzanwendungen einfacher ist Änderungen an alle Arbeitsplätzen zu verteilen.

Ich stelle hier ein Lösung/Möglichkeit dafür zur Verfügung. Dazu habe ich im [Tfr]Designer einige Änderungen durchgeführt.

DER DESIGNER

Der Designer hat nun einige Eigenschaften (Properties) mehr:

AskBeforeDBSave:

Wenn der Wert True ist wird vor jedem Speichern in die Datenbank gefragt ob die Änderungen gespeichert werden sollen. YES: es wird gespeichert – NO: es wird nicht gespeichert. Wenn dieser Wert False ist wird der Report in der Datenbank beim Aufruf von Speichern ohne Nachfrage gespeichert.

DataField:

Das Bezeichnung des Feldes, in das der jeweilige Report gespeichert wird. Bei Firebird ist das ein Feld vom Typ BLOB SUB_TYPE BINARY. D.h. der Report wird im Binärformat in der Datenbank gespeichert. Im beigelegten Beispiel ist das das Datenfeld REPORTSREPORT. Da der Report als XML-Datei gespeichert wird sollte auch ein BLOB SUB_TYPE TEXT CHARACTER Feld verwendet werden können. Andere Datenbanken stellen sicher BLOB-Felder mit gleicher Funktion zur Verfügung.

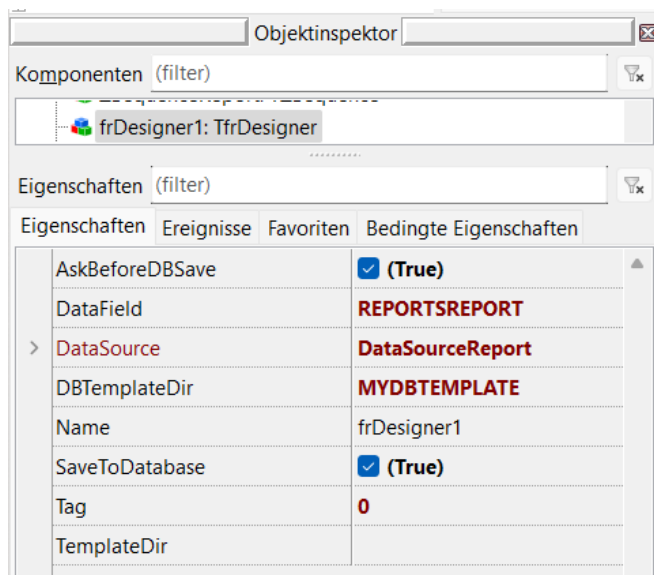
DataSource:

Die Query/Tabelle mittels Datasource mit den Records für die Reports

DBTemplateDir:

Wenn mit der Datenbank gearbeitet wird und hier ein Verzeichnis eingetragen ist kann man mit Speichern unter den jeweiligen Report in dieses Verzeichnis sichern. Bei einer einfachen Verzeichnisangabe liegt dieses Verzeichnis im Verzeichnis des Programmes, mit Laufwerksangabe (?): im jeweils dort angegebenen Verzeichnis. Für andere Betriebssysteme außer Windows müsste noch eine Anpassung gemacht werden. Die Reports werden im standardmäßig verwendeten XML-Format abgespeichert. Es bietet damit die Möglichkeit in einem extra Verzeichnis Vorlagen für Datenbankreports anzulegen.

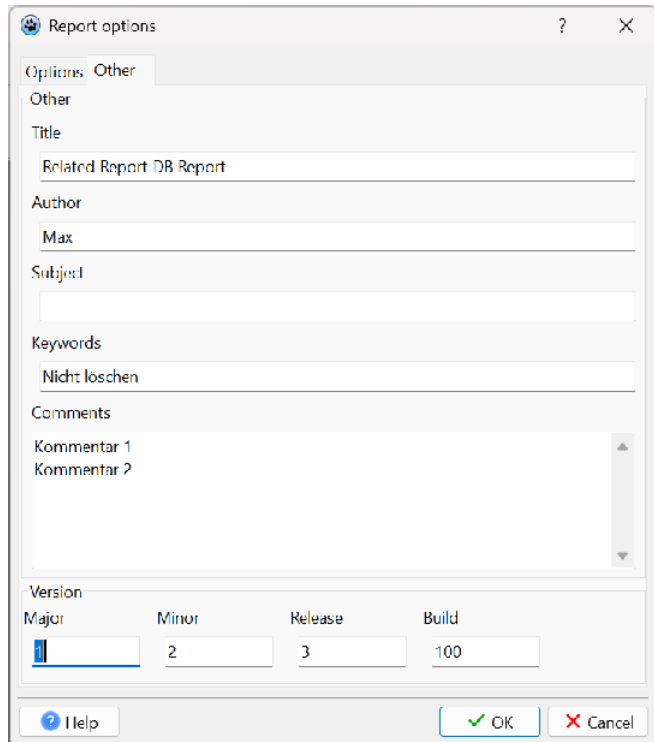
SaveToDatabase: Auch wenn der Designer mit einer Datenbank verbunden ist, aber SaveToDatabase auf False gesetzt ist arbeitet der Designer nicht mit der Datenbank, sondern wie gewohnt mit Dateien.



Wenn im Record die untenstehenden Felder angelegt werden dann interagieren diese mit den Angaben bei im Menü File->Report Options..., Reiter Other und geänderte Daten werden zwischen Report und Datensatz übergeben. Der Text muss genau passen, Groß- oder Kleinschreibung braucht nicht beachtet zu werden. Werden diese Felder nicht angelegt werden diese Daten nur im Report gespeichert. Zur Identifizierung/Überprüfung des jeweiligen Reports ist die Verwendung der Datenfelder vorteilhaft.

Die Felder:

REPORTSTITLE =Title, REPORTSID (der automatische Zähler der Records)=Build, REPORTSCOMMENTS=Comments, REPORTSKEYWORDS=Keywords, REPORTSSUBJECT=Subject, REPORTSREPORTAUTOR=Author.



```
(
REPORTSID Bigint, /* BUILD */
REPORTSTITLE Varchar(50) CHARACTER SET UTF8,
REPORTSREPORTAUTOR Varchar(50) CHARACTER SET UTF8,
REPORTSSUBJECT Varchar(120) CHARACTER SET UTF8,
REPORTSKEYWORDS Varchar(120) CHARACTER SET UTF8,
REPORTSCOMMENTS Varchar(1024) CHARACTER SET UTF8,
REPORTSCREATEDATE Varchar(20) CHARACTER SET UTF8,
REPORTSLASTCHANGE Varchar(20) CHARACTER SET UTF8,
REPORTSFILENAME Varchar(70) CHARACTER SET UTF8,
REPORTSEXTRAVARS BLOB SUB_TYPE TEXT CHARACTER SET
UTF8,
REPORTSREPORT BLOB SUB_TYPE BINARY,
PRIMARY KEY (REPORTSID)
)
```

Im Designer wird bei Aufruf desselben der im aktuellen Datensatz gespeicherte Report übernommen, ein „blättern“ in den Reports im Designer ist nicht vorgesehen und würde auch kontraproduktiv sein, da ja der Designer mit den richtigen Variablen und Datenbanktabellen interagieren muss. Die Übernahme erfolgt im Activate-Ereignis des Reportdesigners. Das hat sich angeboten weil der Designer ja im Modal-Modus angezeigt wird. Wer hier eine bessere Idee hat – bitte melden.

Wenn der Report geändert wurde und der Designer geschlossen wird, wird abgefragt, ob man den Designer in der Datenbank speichern, nicht speichern oder schließen des Designers abbrechen möchte.

Länderspezifische Texte der neu hinzugekommenen Texte zu generieren ist noch nicht implementiert. Ich warte vorerst ab ob diese Erweiterung von den Entwicklern übernommen wird.

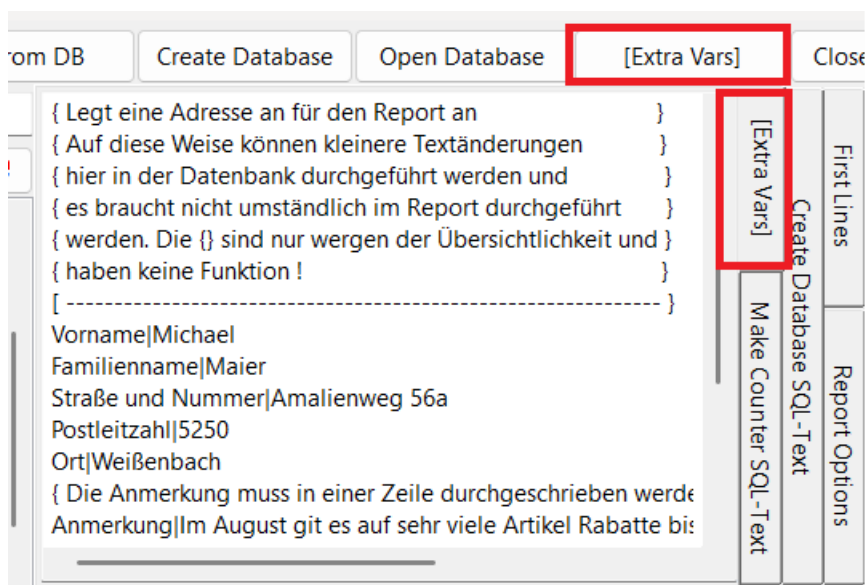
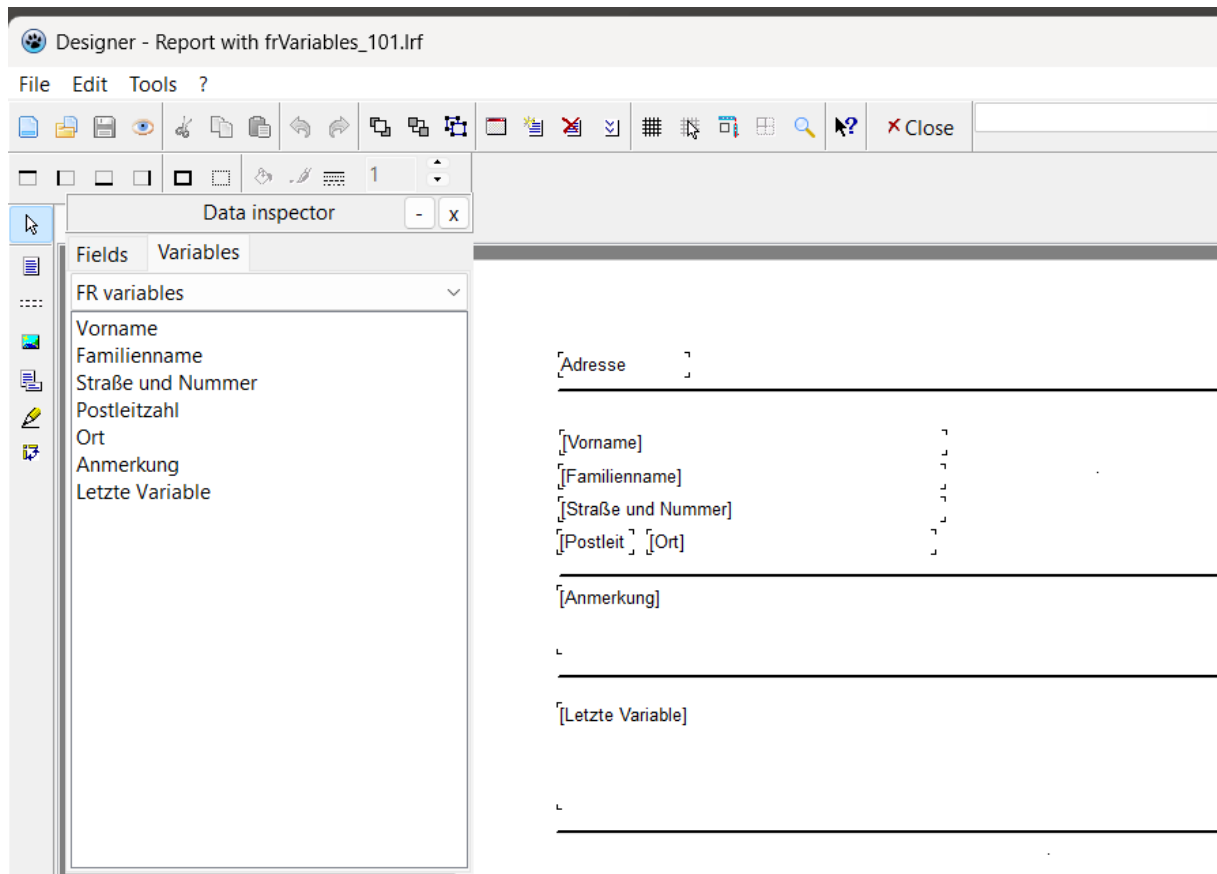
Meine Änderungen im Designer sind mit { --- MAXL ---- } jeweils am Anfang und am Ende gekennzeichnet.

DAS BEISPIELPROGRAMM

Das Beispielprogramm verwendet ZEOS und Firebird. Das Anlegen der Datenbank nebst Beispieldaten kann mit dem Create Database durchgeführt werden, aber wenn keine REPORTS.FDB vorhanden ist wird diese automatisch angelegt. Dem Beispielprogramm ist schon eine Datenbank mit einigen Beispielen beigelegt. Diese kann unter Firebird 4 und Firebird 5 verwendet werden. Frühere Versionen und andere Datenbanken sind dank ZEOS ganz einfach über den Parameter Protocol einstellbar . Und für das Erstellen der entsprechenden Tabellen samt Daten sind alle notwendigen SQL-Dateien in den Reitern als Memo beigelegt. ZEOS kann mit dem Online Package Manager

einfach und bequem installiert werden. Die Datenbank Firebird 5 wird beim Beispielpogramm als embedded Version mitgeliefert, beim Programmstart zugewiesen und braucht nicht installiert zu werden.

Im Record ist ein BLOB SUB_TYPE TEXT CHARACTER Feld mit der Bezeichnung **REPORTSEXTRAVARS** angelegt. Im Datensatz mit der ReportID 101 und dem Report-Titel „Report with frVariables“ sind hier Einträge vorhanden die im Report **als frVariables** zur Verfügung stehen. Es soll damit gezeigt werden dass reportabhängige Daten in der Datenbank passend zum jeweiligen Report gespeichert werden können. Der Report Record bzw. Datensatz kann ja beliebig erweitert werden, z.B. für Bilder. Oder auch eine Steuerung der zu verwendenden Datenbanktabellen über Parameter ist möglich, etc. Diese Daten stehen dann für den jeweiligen Report passend zur Verfügung. Sind diese für den Report notwendigen Daten einmal in der Datenbank angelegt steht einer komfortablen Report-Verwaltung nicht mehr im Wege.



Adresse

Michael

Maier

Amalienweg 56a

5250 Weißbach

Im August git es auf sehr viele Artikel Rabatte bis zu 30%!

Variablen werden nur angenommen wenn in der Zeile ein | Zeichen als Trenner steht!

DIE BUTTONS OBEN

SaveToDatabase und **AskBeforeSave** setzen den Wert in den entsprechenden Checkboxes bei den oben beschriebenen Designer Eigenschaften.

Open Designer öffnet den Designer mit dem in der Datenbank gespeicherten Report

Report from DB zeigt wie der Report im frReport ausgeführt wird.

frReport wurde nicht verändert, die dort vorhandene Procedure zm Laden eines Reports von einem Stream wird zum Laden des Reports verwendet. Eine kleine Procedure ermöglicht die Überweisung des Reports an frReport mittels eines Procedure Aufrufs. frReport1 und die Datenbank in welcher die Reports gespeichert sind werden als Parameter übergeben.

Create Database erstellt die die verwendete Datenbank.

Open Database Öffnet die Datenbank [neu] wenn notwendig.

[Extra Vars] Ruft den Reiter [Extra Vars] auf.

DIE REITER

First Lines:

Zeigt die ersten 2048 Zeichen der XML Datei und kann zur Identifizierung des im aktuellen Record zugrunde liegenden Reports verwendet werden. Da in Memo angezeigte Daten wegen der möglichen Größe einen Fehler verursachen können und dann nicht angezeigt werden ist diese Kürzung notwendig.

Report Options:

Hier kann Text zur Identifizierung des Reports erfasst werden. Sollten die Feldbezeichnungen wie weiter vorne angegeben verwendet werden so interagieren diese Infos mit denen im Report.

Adresses:

Die Beispieldaten der Adresstabelle und weiterer Tabellen sind hier als SQL für das Anlegen der Daten vorgehalten.

Create Database SQL-Text:

Die zum Erstellen der Datenbankstruktur notwendigen SQL Informationen sind hier vorgehalten.

Make Counter SQL-Text:

Für das Anlegen der automatischen Zähler notwendige SQL-Text.

Show Connection Parameters:

Zeigt die von der Firebird Datenbank verwendeten Parameter an.

Show Adresslist:

Zeigt die Beispielladressen und zugehörige Kontaktdaten an.

[Extra Vars]:

Liste von Variablen die im Report verwendet werden können (siehe weiter oben zu [Extra Vars]).

Um den Datenbank Designer verwenden zu können müssen untenstehende Dateien mit den mitgelieferten Dateien im Verzeichnis C:\Lazarus\components\lazreport\source\ überschrieben werden.

C:\Lazarus\components\lazreport\source\lr_desgn.lfm

C:\Lazarus\components\lazreport\source\lr_desgn.pas

Wichtig ist, nachdem die Dateien in das Source-Verzeichnis von LazReport kopiert wurden Lazarus mittels des Menüpunktes „Werkzeuge->Kompiliere Lazarus mit Profil: Normale IDE“ neu zu kompilieren. Es werden ansonsten die neuen Eigenschaften des Reportdesigners nicht angezeigt und das Beispielprogramm wird auch nicht geladen.