AUSGABE 01-02/2021

DATENBANK ENTWICKLER

MAGAZIN FÜR DIE DATENBANKENTWICKLUNG MIT VISUAL STUDIO FÜR DESKTOP, WEB UND CO.

TOP-THEMEN:

André Minhorst Verlag

IsDefault: Standardschaltfläche festlegen

Unter WPF haben Sie verschiedene Möglichkeiten, eine Schaltfläche festzulegen, die beim Betätigen der Eingabetaste ausgelöst wird. Die Einfachste ist die über das Attribut »IsDefault« einer Schaltfläche. Stellen Sie dieses auf den Wert »True« ein, wird die normalerweise durch die Schaltfläche ausgelöste Ereignismethode auch beim Betätigen der Eingabetaste ausgelöst. Unter welchen Voraussetzungen das geschieht, zeigt dieser Artikel.

In Formularen oder Fenstern gibt es oft eine Schaltfläche, deren Funktion beim Betätigen der Eingabetaste ausgelöst werden soll. In vielen Fällen hat diese Schaltfläche die Beschriftung **OK** und soll beispielsweise das Fenster schließen.

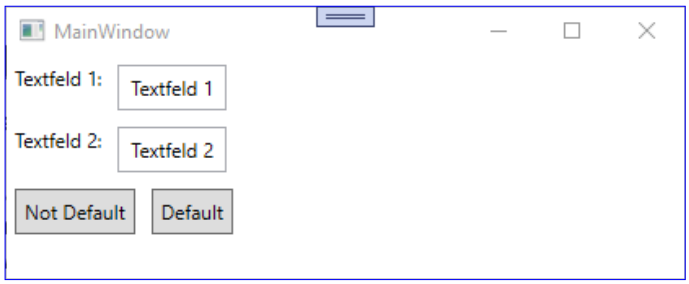


Bild 1: Default-Schaltfläche

Damit eine dafür vorgesehene Schaltfläche dies in einem WPF-Fenster erledigt, hat Microsoft die XAML-Eigenschaft **IsDefault** vorgesehen. Diese stellen Sie für die Schaltfläche, deren Ereignismethode für das Ereignis **Click** beim Betätigen der Eingabetaste ausgelöst werden soll, auf den Wert **True** ein:

```
<StackPanel Grid.Row="2" Orientation="Horizontal" VerticalAlignment="Top">
    <Button x:Name="btnNotDefault" Click="btnNotDefault_Click">Not Default</Button>
    <Button x:Name="btnDefault" IsDefault="True" Click="btnDefault_Click">Default</Button>
</StackPanel>
```

Das Beispiel sieht wie in Bild 1 aus. Wenn danach ein Steuerelement im gleichen Fenster den Fokus hat, das nicht auf das Ereignis **Enter** reagiert, wird die mit **IsDefault** markierte Schaltfläche ausgelöst. Dabei können beispielsweise die folgenden Fälle auftreten:

- Sie betätigen die Eingabetaste, wenn eine andere Schaltfläche oder ein ähnliches Steuerelement, für welches das **Click**-Ereignis implementiert ist, den Fokus hat. In diesem Fall wird die Ereignisprozedur für das Ereignis **Click** der Schaltfläche mit dem Fokus ausgelöst.

- Sie betätigen die Eingabetaste beispielsweise in einem **TextBox**-Steuerelement. Wenn dieses eine Ereignismethode für das Ereignis **Enter** enthält, wird dieses zuerst ausgeführt und danach das **Click**-Ereignis der Schaltfläche mit **IsDefault** = **True**.

- Wenn Sie jedoch für ein **TextBox**-Steuerelement die Eigenschaft **AcceptsReturn** auf **True** einstellen und die Eingabetaste betätigen, wenn dieses Steuerelement den Fokus hat, dann händelt dieses selbst das Betätigen der Eingabetaste, indem einen Zeilenumbruch bewirkt.

IsCancel: Abbrechen-Schaltfläche festlegen

Unter WPF gibt es verschiedene Wege, um zu definieren, was in einem Fenster beim Betätigen der Escape-Taste geschehen soll. Sie könnten beispielsweise die Ereignisse abfangen, die beim Drücken einer Taste ausgelöst werden und dabei prüfen, ob der Benutzer die Escape-Taste gedrückt hat. Meist jedoch gibt es eine Schaltfläche, welche die gleiche Funktion bereithalten soll wie die Escape-Taste – zum Beispiel das Schließen des Fensters über eine Abbrechen-Schaltfläche ohne Speichern der Daten. Wie Sie dies ganz einfach mithilfe des Attributs »IsCancel« erledigen, zeigt der vorliegende Artikel.

Theoretisch könnten Sie die Tastatureingaben des Benutzers abfangen, bis dieser die **Escape**-Schaltfläche betätigt und dann in einer entsprechenden Ereignismethode die gewünschten Anweisungen ausführen – zum Beispiel zum Schließen des aktuellen Fensters mit Verwerfen der getätigten Änderungen.

Bild 1: **Cancel**-Schaltfläche per **Escape** bedienen

Allerdings bietet WPF für das **Button**-Steuerelement ein Attribut, mit dem Sie dies noch einfacher erledigen können – ohne eine Zeile zusätzlichen Code. Während wir im folgenden Beispiel der Schaltfläche **btnOK** den Wert **True** für das Attribut **IsDefault** zuweisen, wodurch die dahinter stehende Ereignismethode beim Verwenden der Eingabetaste ausgelöst wird – mehr zu diesem Attribut siehe Artikel **IsDefault: Standardschaltfläche festlegen** (**www.datenbankentwickler.net/250**), verwenden wir für die Schaltfläche **cmdCancel** das Attribut **IsCancel**. Auch für dieses legen wir den Wert **True** fest. Das bewirkt, dass das Betätigen der **Escape**-Taste durch den Benutzer direkt an die Ereignismethode **Click** dieser Schaltfläche weitergeleitet wird:

```
<StackPanel Grid.Row="2" Orientation="Horizontal" VerticalAlignment="Top">
    <Button x:Name="btnOK" IsDefault="True" Click="btnOK_Click">OK</Button>
    <Button x:Name="btnCancel" IsCancel="True" Click="btnCancel_Click">Cancel</Button>
</StackPanel>
```

Dieser weisen Sie dann die entsprechende Anweisung zu – im folgenden Beispiel einfach nur die Anzeige eines Meldungsfensters:

```
Private Sub btnCancel_Click(sender As Object, e As RoutedEventArgs)
    MessageBox.Show("Cancel")
End Sub
```

Im Gegensatz zur Schaltfläche mit dem Wert **True** für das Attribut **IsDefault** wird die Taste mit **IsCancel** bei jeder Betätigung der **Escape**-Taste ausgelöst, auch wenn der Fokus sich gerade auf einer anderen Schaltfläche befindet.

SQL Server: Fehler bei Entwurfsänderungen

In letzter Zeit habe ich gelegentlich Änderungen an den Tabellen verschiedener SQL Server-Datenbanken vorgenommen. Einige Änderungen dienten der Erweiterung um neue Felder, andere haben Primärschlüssel, Indizes, Beziehungen und andere Elemente angepasst. Die betroffenen Tabellen enthalten teilweise sehr viele Daten, sodass SQL Server recht viel Zeit beanspruchte, die Tabellen zu ändern. Teilweise gab es Fehler, weil Zeitlimits überschritten wurden. Teilweise konnten aber auch neue Beziehungen oder Indizes nicht angelegt werden, weil die Integrität der Daten nicht gegeben war. Dieser Artikel zeigt, welche Einstellungen Sie dann ändern können und welche Ursachen es gibt, wenn all dies nichts hilft.

SQL Server Entwurfsänderungen mit Timeout

Manchmal sind Änderungen am Entwurf einer Tabelle mit sehr vielen Datensätzen für den SQL Server sehr aufwendig durchzuführen. Oft reicht die Zeit nicht aus, weil zum Beispiel eine neue, temporäre Tabelle erzeugt, die Datensätze kopiert und die alte Tabelle gelöscht sowie die neue in die alte umbenannt werden muss. In solchen Fällen erscheint die Meldung aus Bild 1.

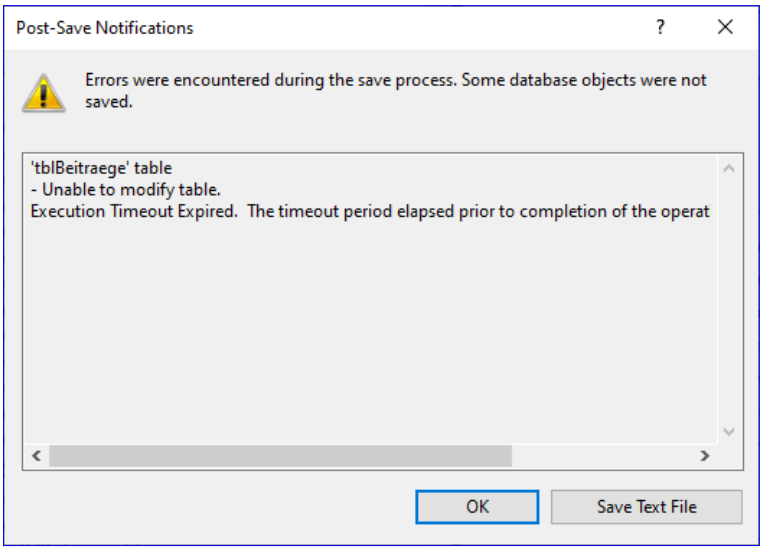


Bild 1: Timeout beim Ändern eines Feldnamens

Das Zeitlimit können Sie über die Optionen des SQL Server Management Studios einstellen. Dazu öffnen Sie den Options-Dialog mit dem Befehl **Tools|Options**. Hier finden Sie unter **Designers|Table and Database Designers** die Option **Override connection string time-out value for table designer updates**, der aktiviert sein sollte, und darunter die Einstellung **Transaction time-out after**, die standardmäßig 30 Sekun-

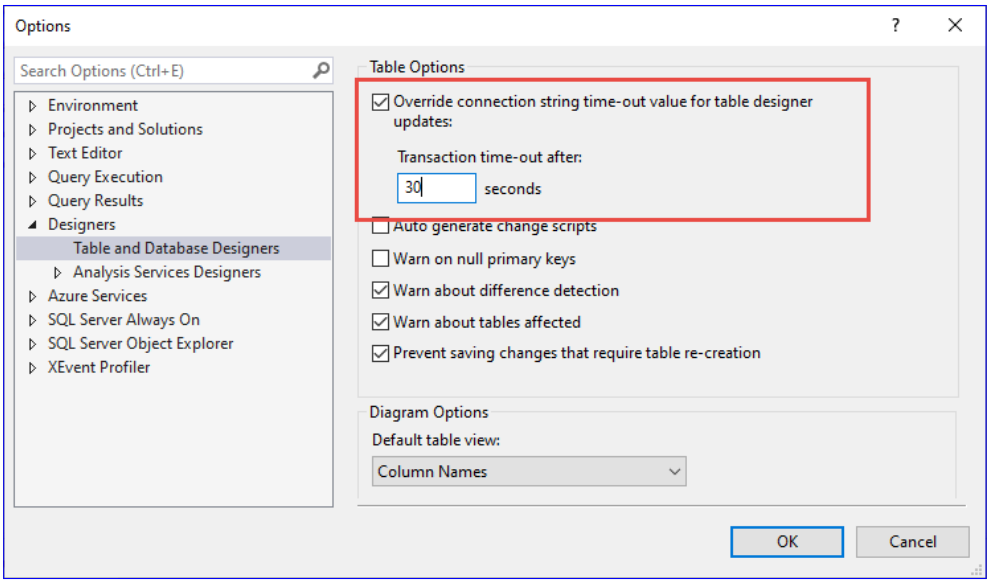


Bild 2: Einstellungen für den Timeout

den beträgt. Wenn Sie also Timeouts haben, setzen Sie einfach diese Einstellung auf einen höheren Wert (siehe Bild 2).

Entwurfsänderungen, die das Neuerstellen der Tabelle erfordern

Wenn Sie eine der folgenden Aktionen im Entwurf einer Tabelle durchführen und diese Änderungen dann speichern wollen, erhalten Sie eventuell einen weiteren Fehler, den Bild 3 zeigt:

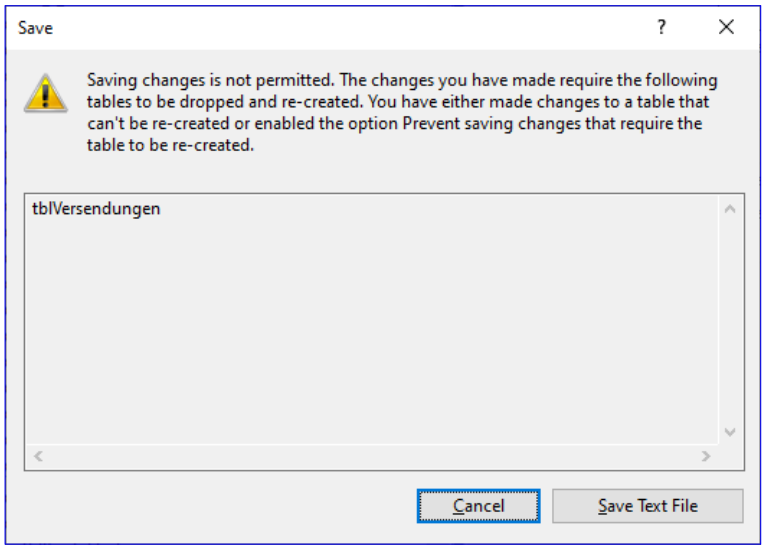


Bild 3: Fehler, weil die Tabelle nicht neu erstellt werden kann

- Sie ändern die Feldeigenschaft **Allow Nulls**.
- Sie ändern die Reihenfolge von Feldern in der Tabelle.
- Sie ändern den Datentyp eines Feldes.
- Sie fügen ein neues Feld hinzu.

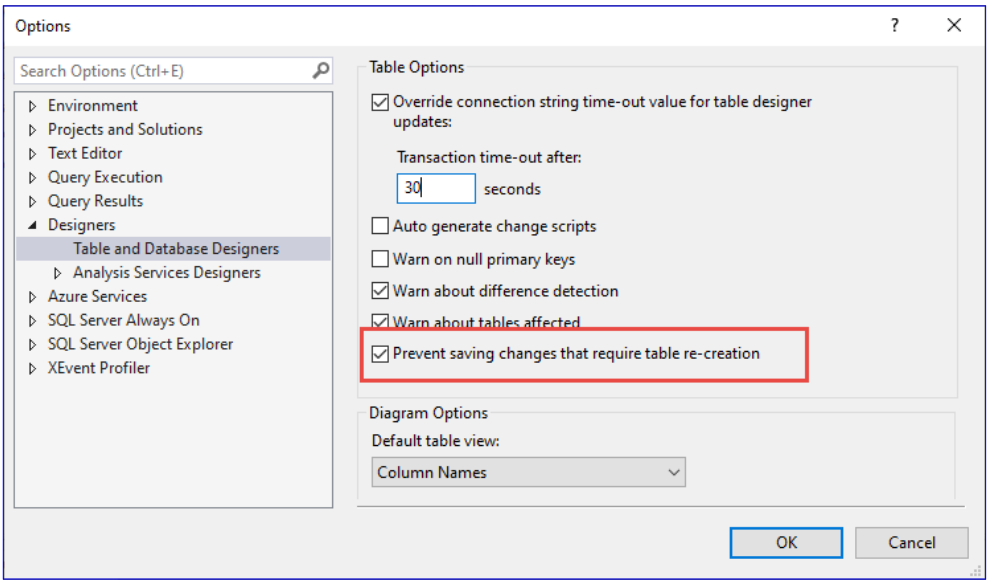


Bild 4: Einstellung zum Freigeben der Neuerstellung von Tabellen beim Ändern

Warum erhalten Sie den Fehler »eventuell«? Weil es davon abhängt, ob die Einstellungen des SQL Servers das Speichern von Änderungen, die ein Neuerstellen der Tabelle erfordern, erlauben. Diese Einstellung finden Sie, wenn Sie im SQL Server Managgement Studio mit Tools|Options den Options-Dialog öffnen. Hier wechseln Sie links zum Bereich **Designers|Table and Database Designers** und finden dort die Option **Prevent saving changes that require table re-creation** (siehe Bild 4). Wenn diese aktiviert ist, können die oben genannten Änderungen nicht gespeichert werden. Um das zu erledigen, müssen Sie die Option deaktivieren.

SQL Server Entwurfsänderungen für Fremdschlüsselfelder

In einer Datenbank habe ich festgestellt, dass eine Beziehung zwischen dem Fremdschlüsselfeld **KundeID** der Tabelle **tblVersendungen** und der Tabelle **tblKunden** noch gar nicht angelegt war. Also habe ich diese nachträg-

lich hinzugefügt. Dazu öffnen Sie die Tabelle mit dem an der Beziehung beteiligten Fremdschlüsselfeld in der Entwurfsansicht, klicken mit der rechten Maustaste auf das Fremdschlüsselfeld und wählen aus dem Kontextmenü den Eintrag **Relationships...** aus (siehe Bild 5). Ein Klick auf **Add** öffnet den Dialog **Foreign Key Relationships**. Hier klicken Sie auf die Eigenschaft **Tables And Columns Specification** und dann auf die Schaltfläche mit den drei Punkten (**...**) und stellen im folgenden Dialog die notwendigen Informationen zusammen. Nach dem Schließen mit **Close** und dem Speichern des geänderten Entwurfs erschien in meinem Fall die Meldung aus Bild 6. Diese lautet vollständig:

```
Die ALTER TABLE-Anweisung steht in Konflikt mit der FOREIGN KEY-Einschränkung "FK_tblVersendungen_tblKunden". Der Konflikt trat in der Kundenverwaltung-Datenbank, Tabelle "dbo.tblKunden", column 'KundeID' auf.
```

Dies deutet darauf hin, dass im Fremdschlüsselfeld **KundeID** der Tabelle **tblVersendungen** Werte enthalten sind, die es im Feld **KundeID** der Tabelle **tblKunden** nicht gibt – es kann also keine Beziehung mit referentieller Integrität hergestellt werden.

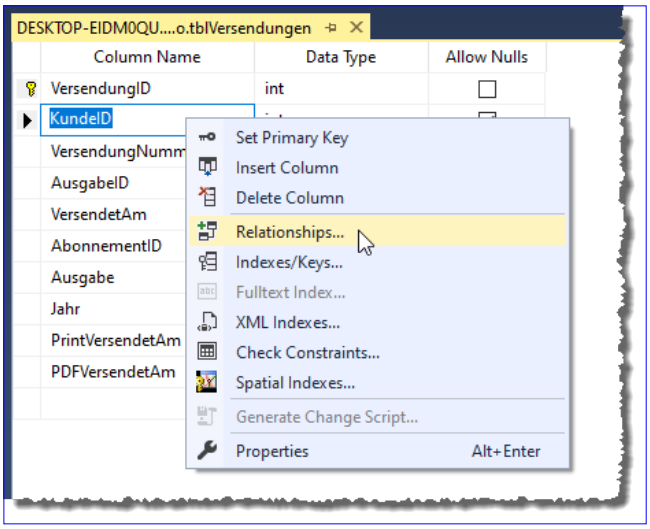


Bild 5: Hinzufügen einer Beziehung

Der Fehler liegt also an den Daten und wir können diesen je nach Menge der »falschen« Daten mehr oder weniger schnell beheben. Wer die Arbeit mit Access gewohnt ist, findet diese Datensätze wohl am schnellsten mit einer geeigneten, über den Abfrageentwurf zusammengeklickten Abfrage. Für die Datensätze, die Probleme bereiten, findet SQL Server keine Entsprechung zu den Werten im Fremdschlüsselfeld im Primärschlüsselfeld der verknüpften Tabelle. Wir erstellen also eine **RIGHT JOIN**-Abfrage für die beiden Tabellen **tblKunden** und **tblVersendungen**, bei der das Primärschlüsselfeld **KundeID** der Tabelle **tblKunden** als Kriterium den Wert **NULL** erhält. Auf diese Weise finden wir in einer neuen Abfrage im SQL Server Management Studio schnell die zu überarbeitenden Datensätze (siehe Bild 7). Gegebenenfalls erstellen Sie diese Abfrage auch gleich in der Access-Datenbank, die an diese Tabellen gebunden ist, und untersuchen dort, wie diese Datensätze angepasst

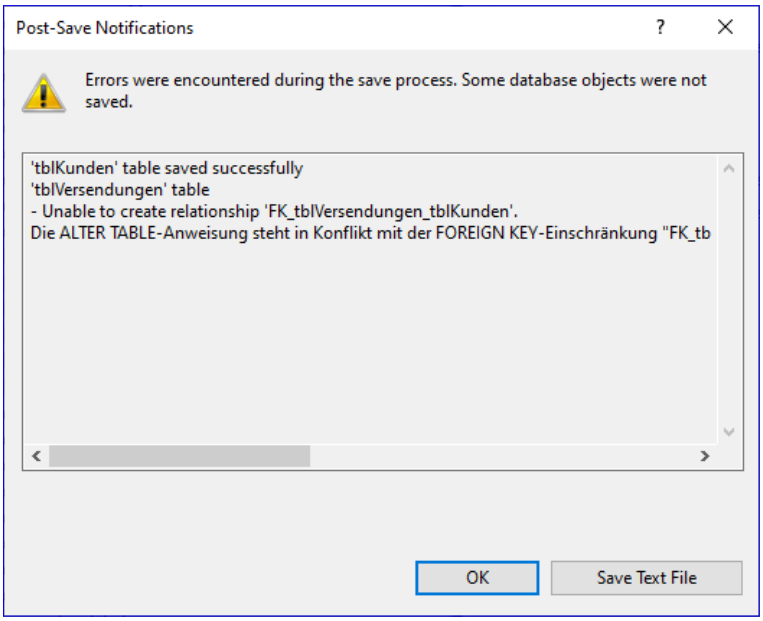


Bild 6: Fehler beim Hinzufügen einer Beziehung

AdventureWorks: Schnelle Beispieldatenbank

Vielleicht benötigen Sie zu irgendeinem Zweck einmal eine SQL Server-Beispieldatenbank. Microsoft stellt eine solche zur Verfügung: Diese heißt AdventureWorks und steht in verschiedenen Versionen zum kostenlosen Download bereit. Dieser Artikel zeigt, wo Sie diese Beispieldatenbank finden und wie Sie diese aus einem Backup auf ihrem Rechner wiederherstellen.

Download der AdventureWorks-Beispieldatenbank

Die **AdventureWorks**-Beispieldatenbank finden Sie zum Download, wenn Sie bei Google **AdventureWorks** eingeben. Hier finden Sie drei verschiedene Versionen mit den Bezeichnungen OLTP (Online Transaction Processing), Data Warehouse und Lightweight. Wir verwenden hier die Lightweight-Version. Die Datenbank kommt außerdem für verschiedene SQL Server-Versionen. Wählen Sie einfach die für Ihre Version des SQL Servers passende Version der Beispieldatenbank aus.

Wir haben so für SQL Server 2019 die Datei **AdventureWorksLT2019.bak** heruntergeladen.

Wiederherstellung der AdventureWorks-Beispieldatenbank

Um die Datenbank aus der Sicherungskopie wiederherzustellen, speichern wir diese zunächst in dem Verzeichnis, in das SQL Server standardmäßig Backup-Dateien ablegt. Dieses Verzeichnis lautet unter Windows 10 beispielsweise wie folgt:

```
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup
```

Nun haben Sie die Möglichkeit, die Datenbank über die Benutzeroberfläche des SQL Server Management Studios wiederzustellen. Dazu öffnen Sie SQL Server Management Studio und melden sich an der SQL Server-Instanz an, in der Sie die Datenbank wiederherstellen möchten.

Wiederherstellen per Benutzeroberfläche

Anschließend klicken Sie mit der rechten Maustaste auf den Eintrag **Databases** des Objekt-Explorers des SQL Server Management Studios. Hier finden Sie den Kontextmenü-Befehl **Restore Database...** und betätigen diesen (siehe Bild 1).

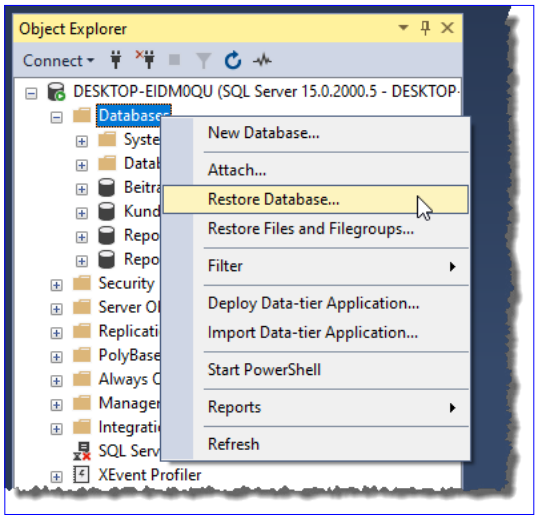


Bild 1: Kontextmenü-Eintrag zum Wiederherstellen einer Datenbank

Im nun erscheinenden Dialog **Restore Database** wählen Sie im rechten Bereich unter **Source** die Option **Device** aus und klicken auf die rechts daneben befindliche Schaltfläche mit den drei Punkten (**...**).

Danach erscheint der Dialog **Select backup device**, in dem Sie wiederum über die Schaltfläche **Add** den Dialog zum Auswählen der Backup-Datei auswählen (siehe Bild 2).

EDM mit vorhandener SQL Server-Datenbank

In einigen der vorherigen Ausgaben haben wir uns mit dem Erstellen eines Entity Data Models auf Basis von Access-Datenbanken konzentriert. Dabei haben wir aus dem Datenmodell der Access-Datenbank ein Entity Data Model erstellt, mit dem dann die SQL Server-Datenbank angelegt werden konnte. Nun wollen wir den vermutlich nicht so seltenen Fall betrachten, dass bereits eine SQL Server-Datenbank existiert. Dabei zeigen wir, wie Sie das Entity Data Model erstellen und damit auf die in der SQL Server-Datenbank enthaltenen Daten zugreifen können – und wie Sie die Namen von Entitäten und Eigenschaften angleichen, wenn die in den Tabellen und Feldern verwendeten Namen nicht passen.

EDM manuell oder per Assistent erstellen?

Bei unseren Techniken zum Erstellen eines Entity Data Models auf Basis des Datenmodells einer Access-Datenbank haben wir gesehen, dass der entstehende Code überschaubar und von der Anzahl der Tabellen und Felder abhängig ist. Je nachdem, ob die Anwendung alle Tabellen des Datenmodells der Datenbank verwenden soll oder vielleicht nur einige wenige, kann es eine Alternative sein, die Objekte manuell zu erstellen.

Ein wichtiger Faktor dabei ist, ob die Tabellen und Felder so benannt sind, dass Sie diese direkt in das Entity Data Model übernehmen können. Dabei ist es zum Beispiel eher untypisch, eine Entität mit **tblKunde** zu bezeichnen, auch wenn die Tabelle **tblKunden** heißt. Schöner wäre es, wenn wir hier **Kunde** statt **tblKunden** verwenden könnten. Wie wir gleich sehen werden, übernimmt Visual Studio bei Verwendung des Assistenten immer die Namen der Tabellen und Felder für die Entitäten und ihre Eigenschaften. Wenn wir also alle oder auch nur einige Tabellen und ihre Felder als Grundlage für die Erstellung eines Entity Data Models verwenden, müssen wir noch

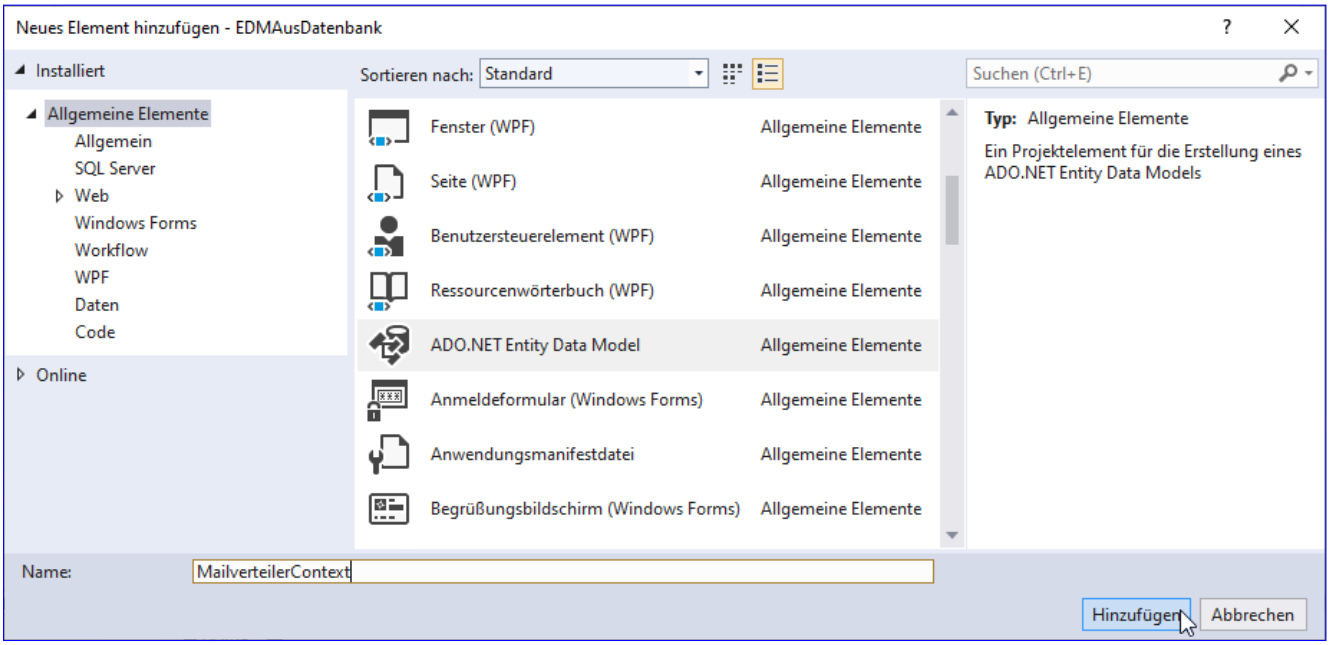


Bild 1: ADO.NET Entity Data Model hinzufügen

einen Weg finden, wie wir eine Art Mapping erstellen. Dieses soll dafür sorgen, dass wir die Tabellen- und Feldnamen wie **tblKunden** oder **KundeID** in der Datenbank beibehalten und gleichzeitig Entitätsnamen wie **Kunde** oder eine Bezeichnung wie **ID** für ein Schlüsselfeld verwenden können.

Entity Data Model per Assistent erstellen

Wenn wir das Entity Data Model per Assistent erstellen, legen wir dazu zunächst ein neues Projekt des Typs **WPF-App (.NET Framework)** für Visual Basic an. Danach fügen wir ein neues Element des Typs **ADO.NET Entity Data Model** zum Projekt hinzu. Dazu wählen Sie den Menübefehl **Projekt|Neues Element hinzufügen...** aus und wählen im nun erscheinenden Dialog den Eintrag **ADO.NET Entity Data Model** aus. Außerdem legen Sie unter **Name** die gewünschte Bezeichnung fest, in diesem Fall **MailverteilerContext** (siehe Bild 1). Der Hintergrund ist, dass wir in einem weiteren Artikel in der folgenden Ausgabe eine Anwendung mit einem Mailverteiler programmieren, der nur wenige Tabellen der SQL Server-Datenbank nutzen soll – und dessen Entity Data Model wir hier gleich vorbereiten.

Danach erscheint der Dialog **Assistent für Entity Data Model**, in dem wir den Eintrag **Code First aus Datenbank** auswählen (siehe Bild 2).

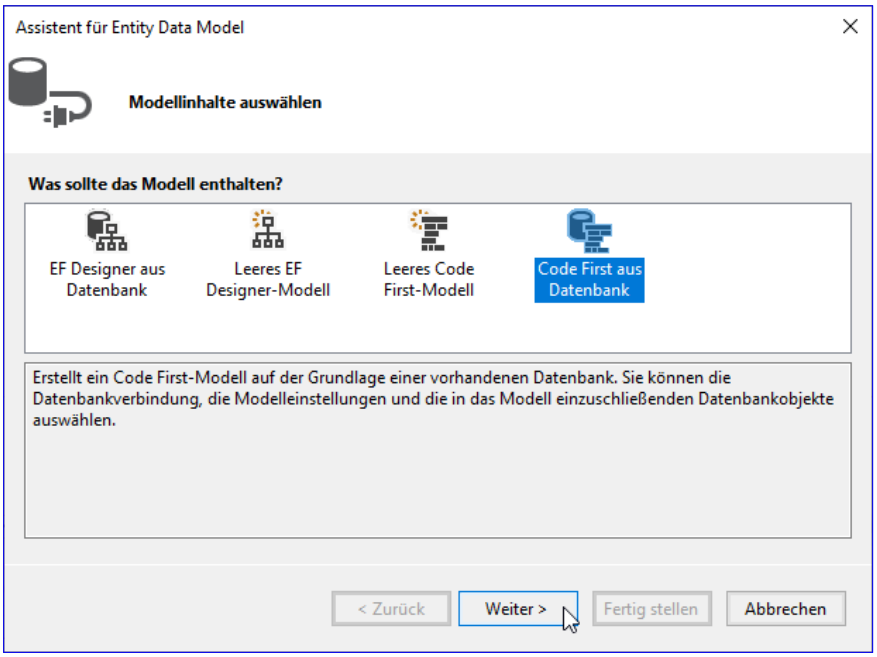


Bild 2: Auswahl des Eintrags **Code First aus Datenbank**

Verbindung einstellen

Im nächsten Schritt klicken Sie im Dialog **Wählen Sie Ihre Datenverbindung aus** auf

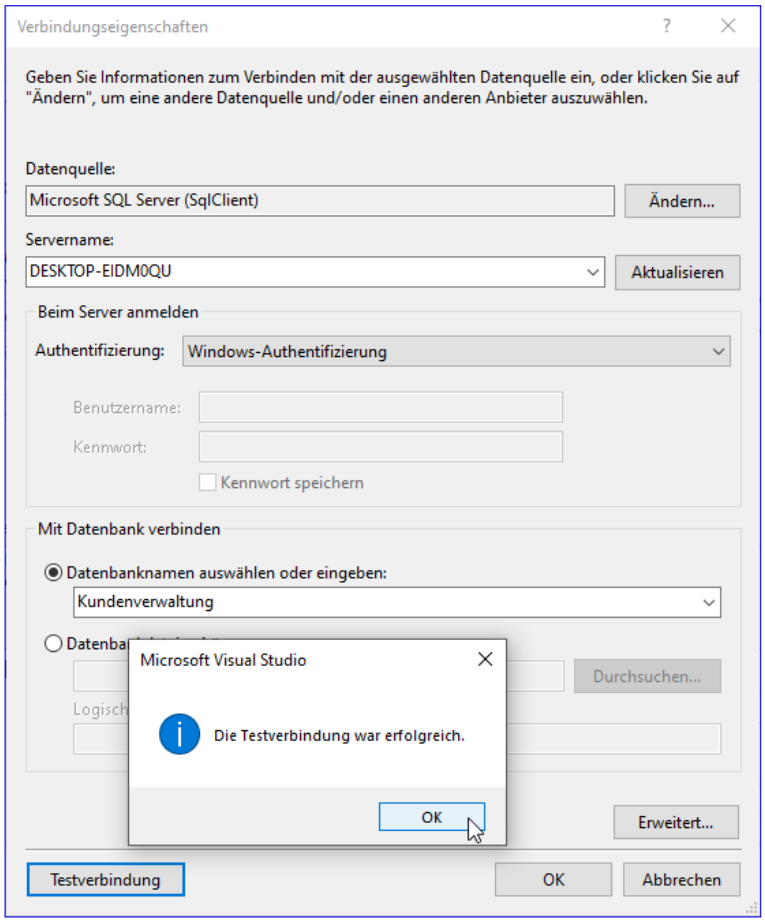


Bild 3: Einstellen der Verbindungseigenschaften

den Befehl **Neue Verbindung...** (außer, Sie haben dort zuvor schon einmal die gewünschte Verbindung angegeben – dann können Sie diese direkt auswählen). Damit erscheint der Dialog **Verbindungseigenschaften**, wo Sie zuerst den Namen des SQL Servers eingeben oder diesen aus dem SQL Server Management Studio herauskopieren und einfügen, da das Einlesen gelegentlich etwas länger dauert. Wenn Sie den Servernamen eingegeben haben, stellen Sie noch die Authentifizierungsart ein und hier gegebenenfalls noch die Zugangsdaten. Außerdem können Sie nun die Datenbank auswählen. Nach einem Test mit einem Klick auf die Schaltfläche **Testverbindung** und der Bestätigung, dass die Verbindung funktioniert, beenden Sie den Dialog mit **OK** (siehe Bild 3).

Verbindungszeichenfolge prüfen

Nach dem Schließen kehren wir zum Dialog-Schritt **Wählen Sie Ihre Datenverbindung aus** zurück. Dieser zeigt Ihnen nun die zusammengestellte Verbindungszeichenfolge an. Außerdem können Sie hier festlegen, unter welchem Namen diese Einstellungen in der Datei **App.Config** hinzugefügt werden soll (siehe Bild 4).

Datenbankobjekte auswählen

Der nächste Schritt bietet die einzelnen Tabellen und Views der

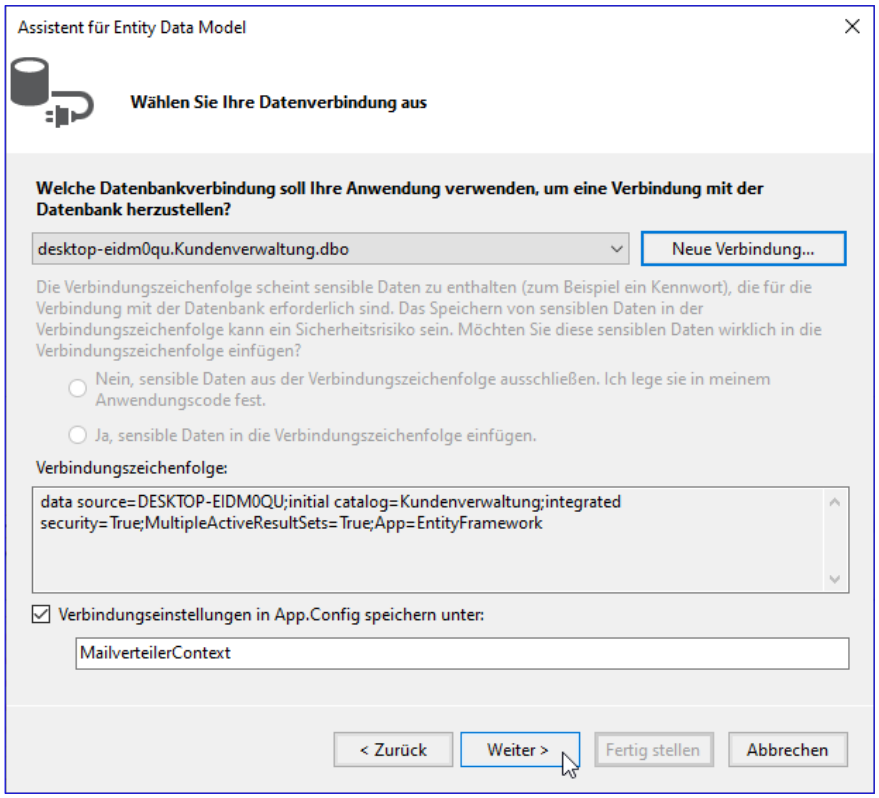


Bild 4: Einstellen weiterer Informationen

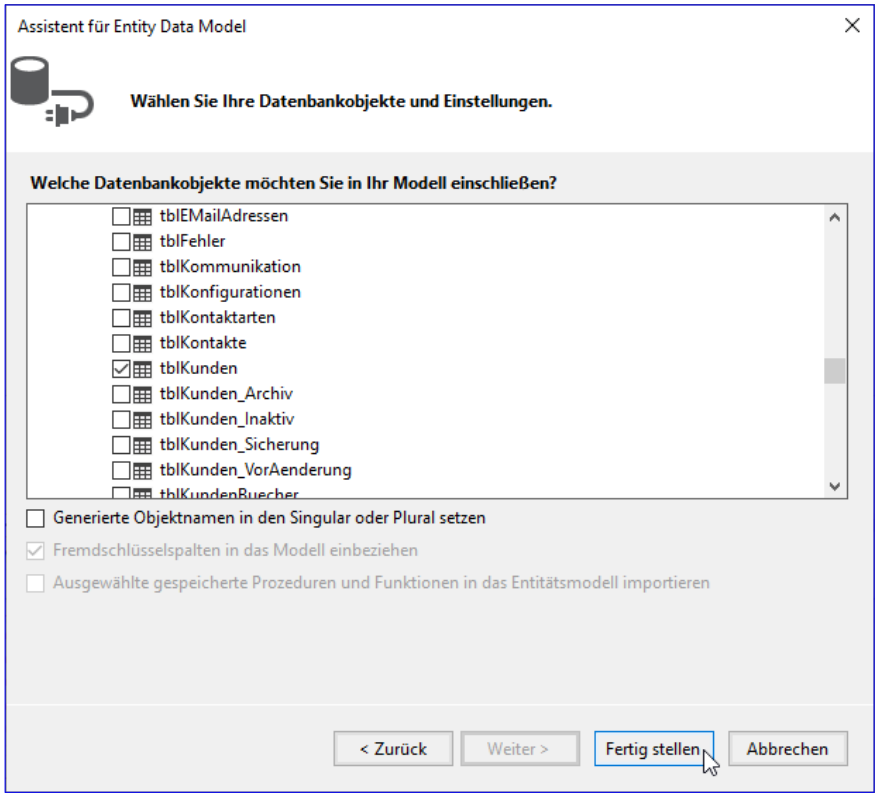


Bild 5: Auswahl der zu verwendenden Tabellen

SQL Server-Datenbank zur Auswahl an. Hier selektieren wir zuerst einmal die beiden Tabellen **tblAnreden** und **tblKunden** (siehe Bild 5). Die Option **Generierte Objektnamen in den Singular oder Plural setzen** sorgt dafür, dass für die Ermittlung der **DbSet**-Elemente, also der Auflistungselemente für die Entitäten, jeweils noch ein **s** an den Tabellennamen angehängt wird. Für die Tabelle **tblKunden** würde der Assistent dann ein **DbSet** namens **tblKundens** erstellen. Das ist noch weniger funktional, als die **DbSet**-Elemente und die Entitäten gleichermaßen mit **tblKunden** zu benennen.

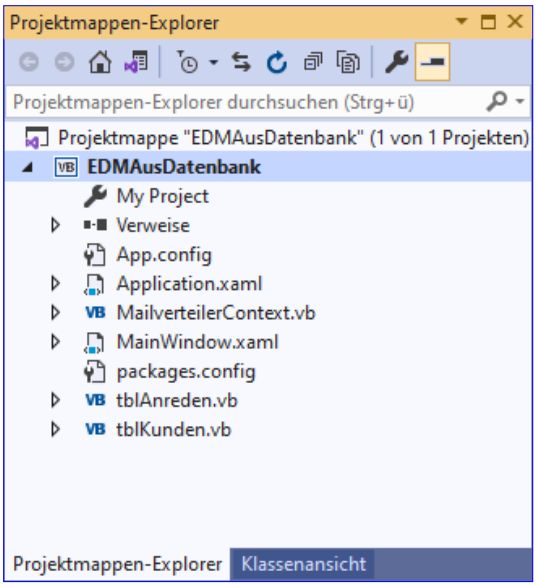


Bild 6: Die neuen Objekte im Projektmappen-Explorer

Erstellen des Entity Data Models

Nach einem Klick auf die Schaltlfläche **Fertigstellen** arbeitet Visual Studio ein wenig und kurz darauf erscheinen einige neue Einträge im Projektmappen-Explorer beziehungsweise wurden angepasst (siehe Bild 6). Die neuen Objekte heißen:

- **MailverteilerContext.vb**: Enthält die Definition der **DbSet**-Elemente sowie eine Methode namens **OnModelCreating**, die weitere Eigenschaften für verschiedene Felder der Tabelle **tblKunden** einstellt.
- **tblAnreden.vb**: Entitätsklasse für die Tabelle **tblAnreden**
- **tblKunden.vb**: Entitätsklasse für die Tabelle **tblKunden**

Schließlich finden wir in der Datei **App.config** noch ein paar neue Zeilen, welche das Entity Framework definieren sowie die Angabe der Verbindungszeichenfolge:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  ...
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
    ...
  </entityFramework>
  <connectionStrings>
    <add name="MailverteilerContext" connectionString="data source=DESKTOP-EIDM0QU;initial catalog=Kundenverwaltung;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

Code First Mapping per Fluent API

Wenn Sie ein Code First-Entity Data Model entwerfen, gibt es bestimmte Konventionen, die standardmäßig greifen. So heißen Entitäten wie der Singular der zugrunde liegenden Tabellennamen oder der Primär- und Fremdschlüssel werden aus Feldern abgeleitet, welche die Zeichenkette ID enthalten – gegebenenfalls kombiniert mit dem Entitätsnamen. Wenn Sie mit einem Datenmodell daherkommen, dessen Tabellen das Präfix »tbl« mitbringen, wollen Sie dieses nicht in den Entitätsnamen wiederfinden und gegebenenfalls möchten (oder müssen) Sie auch noch Feldnamen ändern und diese anschließend mappen. Eine Möglichkeit dazu finden Sie in den Methoden der Fluent Api, die wir in diesem Artikel beschreiben.

Warum Mapping zwischen Datenmodell und Entitäten?

Wenn Sie ein Entity Data Model auf Basis des Datenmodells einer Datenbank erstellen oder andersherum, ist immer eine bestimmte Art von Mapping erforderlich. Im Optimalfall sind die Tabellen- und Feldnamen genau so eingestellt, dass diese den DbSet- und Entitätsnamen entsprechen.

Sobald Sie aber ein Datenmodell erhalten, das beispielsweise Tabellennamen mit dem Präfix **tbl** (wie **tblArtikel**) enthält und wo die Feldnamen vielleicht nicht den Wünschen des Entwicklers für die Eigenschaften der Entitäten entsprechen, müssen Sie diese auf irgendeine Weise anpassen. Das Datenmodell anzupassen wäre toll, aber in manchen Fällen greifen noch andere Anwendungen auf die Datenbank zu, sodass Sie die Namen von Tabellen und Feldern nicht einfach ändern können.

Also stellt das Entity Framework Möglichkeiten bereit, das Mapping zwischen den Tabellen und Feldern auf der einen und den DbSets und Eigenschaften auf der anderen Seite anzupassen.

Dabei kann das Mapping für verschiedene Zwecke genutzt werden:

- Wenn Sie ein Entity Data Model auf Basis einer bestehenden Datenbank etwa vom SQL Server für Code First erstellt haben und die Auflistungen, Entitätsklassen und ihre Eigenschaften so anpassen wollen, dass es Ihren Programmierkonventionen entspricht, auch wenn die Definition der Tabellen und Felder anders lautet oder

- wenn Sie ein Entity Data Model als Code First-Model programmiert haben und dann per Migration-Funktionalität daraus die Datenbank erzeugen wollen. Wie dies gelingt, schauen wir uns an einigen Beispielen in den folgenden Abschnitten im Detail an.

Mapping mit der Fluent API

Die Fluent API des Entity Frameworks bietet Methoden, die genau an einer Stelle innerhalb des Entity Data Models ausgeführt werden: in der Überschreibung der Methode **OnModelCreating** der **DbContext**-Klasse. Entity Framework ruft diese Methode beim Erstellen des Entity Data Models auf und führt die dort angegebenen Anweisungen zum Anpassen des Mappings aus.

Beispielprojekt

Das Beispielprojekt erstellen wir als neues Projekt des Typs **WPF-App (.NET Framework)** für Visual Basic. Damit wir die nun zu erstellenden Elemente in eine Datenbank übertragen können, fügen wir dem Projekt ein neues Element des Typs **ADO.NET Entity Data Model** hinzu und nennen dieses **FluidAPIContext** (siehe Bild 1).

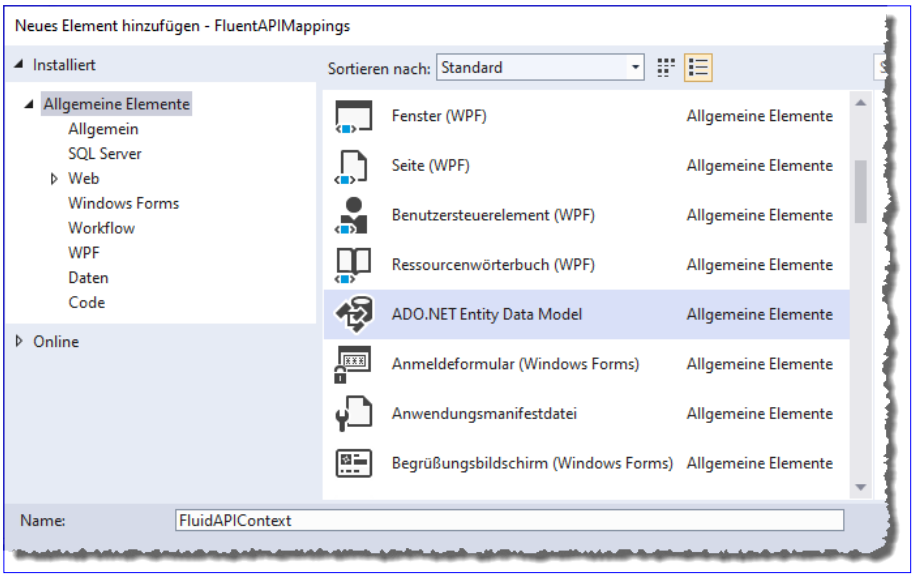


Bild 1: Entity Data Model hinzufügen

Dabei verwenden wir als Vorlage **Leeres Code First-Modell** (siehe Bild 2). Nach einem Klick auf die Schaltfläche **Fertigstellen** erhalten wir im Projektmappen-Explorer ein neues Element namens **FluidAPIContext.vb** sowie ein paar neue Einträge in der Datei **App.config** – zum Beispiel die Verbindungszeichenfolge zum Erstellen beziehungsweise für den Zugriff auf die Datenbank für diese Lösung.

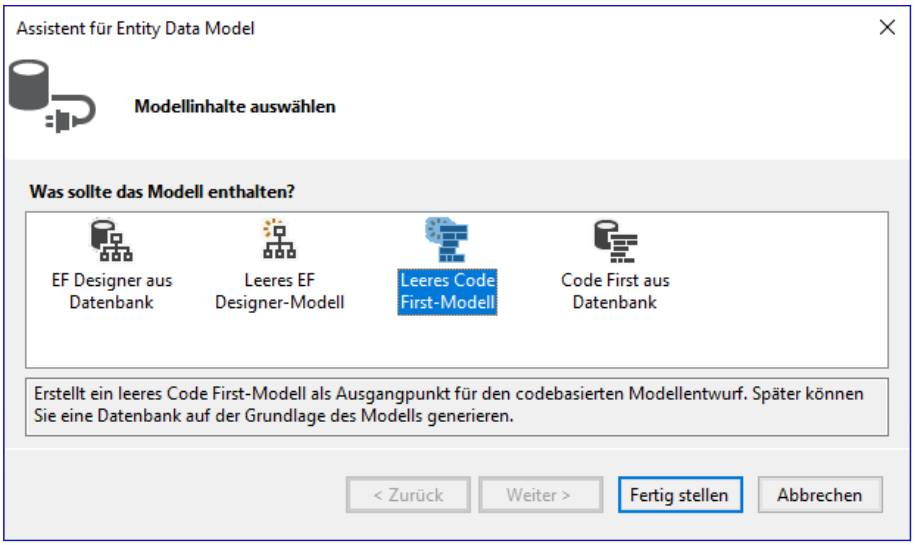


Bild 2: Neues Entity Data Model als leeres Code First-Modell

Einfache Tabelle erstellen

Zu einer einfachen Tabelle gehört ein eindeutiges Primärschlüsselfeld mit Autowert-Funktion sowie die benötigten Felder. Wir beginnen mit einer Entität, auf deren Basis eine Tabelle namens **Anreden** entstehen soll. Diese soll die Felder **ID** und **Name** enthalten. Dazu legen wir eine neue Klassendatei namens **Anrede.vb** an. Der Code der Klasse lautet vorerst:

```
Public Class Anrede
    Public Property ID As Integer
    Public Property Name As String
End Class
```

Für dieses Element fügen wir der **DbContext**-Klasse eine Auflistung hinzu:

```
Public Class FluidAPIContext
    Inherits DbContext
    Public Sub New()
        MyBase.New("name=FluidAPIContext")
    End Sub
    Public Overridable Property Anreden As DbSet(Of Anrede)
End Class
```

Um daraus die Datenbank mit der ersten Tabelle zu erstellen, benötigen wir die Paket-Manager-Konsole, die wir mit dem Befehl **Extras|NuGet-Paketmanager|Paket-Manager-Konsole** anzeigen. In der Konsole setzen Sie nacheinander die folgenden Anweisungen ab:

- **Enable-Migrations**: Aktiviert die Migrationen vom Entity Data Model zu einer SQL Server-Datenbank. Dies fügt den Ordner **Migrations** mit der Datei **Configuration.vb** zum Projekt im Projektmappen-Explorer hinzu.

- **Add-Migration Init**: Erstellt die Befehle für die initiale Migration, die beispielsweise in einer Datei namens **202103111053588_Init.vb** im Ordner **Migrations** angelegt wird. Die hier gespeicherten Befehle werden für das Erstellen beziehungsweise Anpassen der Datenbank verwendet.

- **Update-Database**: Führt die in der mit **Add-Migration** erstellten Klasse enthaltenen Befehle aus, um die Datenbank erstmalig zu erstellen und die angegebene Tabelle anzulegen.

Dies erstellt die neue Datenbank und eine Tabelle, die wir uns im SQL Server Management Studio anschauen können (den Zielserver und den Namen der Datenbank können Sie der Verbindungszeichenfolge in der Datei **App.config** entnehmen). Das Ergebnis sieht wie in Bild 3 aus.

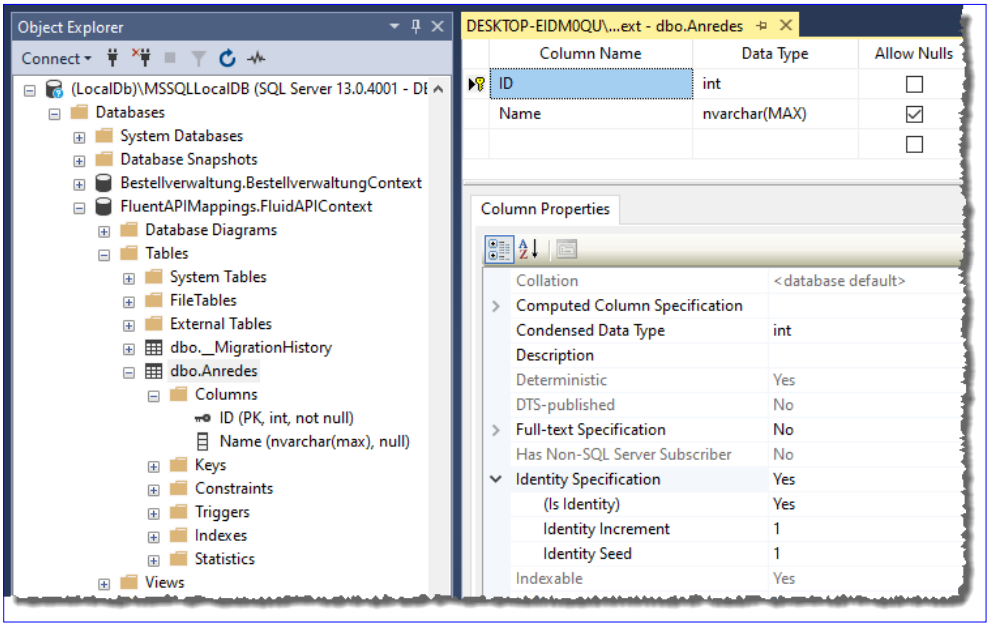


Bild 3: Die neue Tabelle in der frisch erzeugten Datenbank

Eingesetzte Konventionen

Hier erkennen wir gleich, dass bei der Erstellung der Tabelle einige Konventionen umgesetzt wurden. So hat Entity Framework automatisch erkannt, dass die Eigen-

schaft **ID** wohl als Primärschlüsselfeld mit Autowertfunktion verwendet werden soll. Außerdem hat es das Feld **Name** als Feld mit dem Datentyp **nvarchar(MAX)** angelegt, der Nullwerte erlaubt. Als Tabellenname hat Entity Framework aus der Entitätsklasse namens **Anrede** den Namen **Anredes** erzeugt. Dies resultiert ebenfalls aus einer Konvention, die dafür sorgt, dass Entity Framework den Tabellenamen aus dem Namen der Entitätsklasse plus dem **s** für den englischen Plural zusammenstellt.

Umgehen der Konventionen

In den folgenden Abschnitten schauen wir uns an, wie wir diese Konventionen durch Verwendung von Elementen der FluentAPI für das Entity Framework umgehen und mit eigenen Konventionen ersetzen können. Wir wollen zunächst die folgenden Dinge ändern:

- Der Tabellenname soll **Anreden** lauten oder gegebenenfalls auch **tblAnreden**.
- Der Felddatentyp des Feldes **Name** soll **nvarchar(50)** lauten.
- Das Primärschlüsselfeld soll den Namen **AnredeID** erhalten.
- Das Feld **Name** soll in der Tabelle den Namen **Anrede** erhalten.

Anpassen des Tabellennamens

Als Erstes setzen wir die Konvention außer Kraft, die dafür sorgt, dass die zu erstellende Tabelle den Namen der Entitätsklasse plus dem Plural-s als Bezeichnung erhält. Dazu fügen wir der Klasse **FluentApiContext** die Methode **OnModelCreating** hinzu und legen dort eine Anweisung an, welche die Konvention aufhebt:

```
Public Class FluidAPIContext
    Inherits DbContext
    ...
    Public Overridable Property Anreden As DbSet(Of Anrede)

    Protected Overrides Sub OnModelCreating(ByVal modelBuilder As DbModelBuilder)
        modelBuilder.Conventions.Remove(Of PluralizingTableNameConvention)()
    End Sub
End Class
```

Um den Typ **PluralizingTableNameConvention** zu nutzen, fügen wir einen weiteren Namespace zu der Klassendatei hinzu:

```
Imports System.Data.Entity.ModelConfiguration.Conventions
```

Nun aktualisieren wir das Datenmodell, indem wir die folgenden beiden Anweisungen in der Paket-Manager-Konsole aufrufen:

```
Add-Migration Init
Update-Database
```

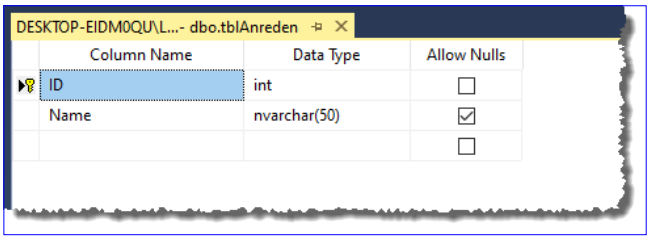


Bild 4: Angepasste Feldgröße

Danach finden wir die Tabelle in der Datenbank unter dem Namen **Anrede** vor. Wir möchten allerdings den Namen **Anreden** oder auch **tblAnreden**. Dazu fügen wir der Methode **OnModelCreating** eine weitere Anweisung hinzu:

```
modelBuilder.Entity(Of Anrede)().ToTable("Anreden")
```

Wenn Sie den Namen **tblAnreden** verwenden möchten, nutzen Sie einfach folgende Anweisung:

```
modelBuilder.Entity(Of Anrede)().ToTable("tblAnreden")
```

Um diese Änderungen durchzuführen, rufen Sie immer wieder die beiden Anweisungen in der Paket-Manager-Konsole auf – wir weisen in den folgenden Abschnitten nicht mehr explizit darauf hin. Außerdem müssen Sie, um die Änderungen in der SQL Server Management Konsole zu sehen, das Element **Tables** über den Kontextmenü-Eintrag **Refresh** aktualisieren.

Anpassen des String-Felddatentyps

Für das Feld Name der Entität **Anrede**, welche die Bezeichnung der Anrede aufnehmen soll, benötigen wir kein **nvarchar(MAX)**, **nvarchar(50)** reicht völlig aus. Zum Einstellen der Feldgröße fügen wir der Methode **OnModelCreating** den folgenden Befehl hinzu:

```
modelBuilder.Entity(Of Anrede)().Property(Function(t) t.Name).HasMaxLength(50)
```

Das sieht schon etwas komplizierter aus. Es geht um die Entität **Anrede** (**Entity (Of Anrede)**). Hier legen wir für die Eigenschaft **Name** über das Attribut **HasMaxLength(50)** die Länge **50** fest. Das Ergebnis sehen Sie in Bild 4.

Anpassen der Feldnamen

Nun wollen wir für das Primärschlüsselfeld den Namen **AnredeID** und für das Feld mit der Bezeichnung der Anrede den Namen **Anrede** statt **Name** verwenden. Hier treffen wir auf einen kleinen Konflikt, wenn wir die Tabelle beispielsweise sowohl mit einer Access-Datenbank als Frontendend verwenden wollen als auch in einem Visual Studio-Projekt. Unter Access kann es zu Problemen kommen, wenn wir **Name** als **Feldname** verwenden, unter Visual Studio darf eine Eigenschaft einer Entitätsklasse nicht so heißen wie die Klasse selbst. Also verwenden wir den Feldnamen **Anrede** und mappen diesen für das Visual Studio-Projekt auf die Eigenschaft **Name**. Damit das Feld **Name** in der Tabelle unter dem Namen **Anrede** erscheint, fügen wir die folgende Zeile zur Methode **OnModelCreating** hinzu:

```
modelBuilder.Entity(Of Anrede)().Property(Function(t) t.Name).HasColumnName("Anrede")
```

Code First Mapping per DataAnnotation

Wenn Sie ein Code First-Entity Data Model entwerfen, gibt es bestimmte Konventionen, die standardmäßig greifen. So heißen Entitäten wie der Singular der zugrunde liegenden Tabellennamen oder der Primär- und Fremdschlüssel werden aus Feldern abgeleitet, welche die Zeichenkette ID enthalten – gegebenenfalls kombiniert mit dem Entitätsnamen. Wenn Sie mit einem Datenmodell daherkommen, dessen Tabellen das Präfix »tbl« mitbringen, wollen Sie dieses nicht in den Entitätsnamen wiederfinden und gegebenenfalls möchten (oder müssen) Sie auch noch Feldnamen ändern und diese anschließend mappen. Eine Möglichkeit dazu bieten die DataAnnotations, die wir in diesem Artikel beschreiben.

Im Artikel **Code First Mapping per Fluent API** (**www.datenbankentwickler.net/252**) haben wir gezeigt, wie Sie die Entitätsklassen eines Entity Data Models auf die Tabellen eines Datenmodells einer SQL Server-Datenbank mappen können beziehungsweise wie Sie dafür sorgen können, dass die Tabellen, Felder und Beziehungen der Tabellen beim Erstellen einer neuen Datenbank auf Basis von Code First nach Ihren Wünschen erstellt werden können. Dabei haben wir ausschließlich die Methoden der Fluent API verwendet, die in einer speziellen Methode namens **OnModelCreating** angelegt werden. Im vorliegenden Artikel wollen wir eine Alternative dazu vorstellen, nämlich die **DataAnnotations**. Dabei handelt es sich um »Anmerkungen«, die direkt in die jeweiligen Entitätsklassen eingetragen werden. Im Gegensatz zu den Methoden, die wir mit der Fluent API in der Methode **OnModelCreating** hinterlegen, landen die DataAnnotations unmittelbar bei den betroffenen Klassen und Eigenschaften. Wir wollen versuchen, die Mapping-Eigenschaften, die wir im oben genannten Artikel mit der Fluent API vorgenommen haben, mit den DataAnnotations zu reproduzieren. Wir verwenden also die gleichen Entitätsklassen und wollen das gleiche Datenmodell daraus erstellen beziehungsweise mappen wie in diesem Artikel.

Erstellen versus Mapping

In den Beispielen dieses Artikels, in denen wir die Anpassungen von Entity Data Model zu den Tabellen des zu erzeugenden Datenmodells Schritt für Schritt anpassen, erstellen wir die Datenbank einmal neu und führen die nachfolgend beschriebenen Änderungen Schritt für Schritt aus. Das erledigen wir mit drei Methoden, die wir über die Paket-Manager-Konsole absetzen. Diese Methoden lauten:

- **Enable-Migration**: Aktiviert die Migrationen für das Entity Data Model. Dieser Schritt ist nur einmal vor dem ersten Aufruf der nachfolgend beschriebenen Anweisung **Add-Migration** notwendig.

- **Add-Migration <Bezeichnung>**: Fügt eine neue Migration hinzu. Beim ersten Aufruf wird dabei eine Klasse erstellt, die Methoden mit allen Befehlen enthält, um die Datenbank und die enthaltenen Tabellen samt Feldern, Indizes und Beziehungen zu erstellen. Bei Aufrufen nach Änderungen am Entity Data Modell werden zu der jeweils neu erstellten Klasse nur Methoden mit Befehlen hinzugefügt, die für das Anwenden der Änderungen notwendig sind. Die Klassen und die enthaltenen Methoden und Befehle werden durch die nachfolgend vorgestellte Anweisung angewendet.

- **Update-Database**: Erstellt die Datenbank beziehungsweise nimmt die Änderungen vor, die durch **Add-Migration** hinzugefügt wurden.

Beispielprojekt für DataAnnotations

Das Beispielprojekt setzen wir als WPF-App für das .NET Framework mit Visual Basic auf. Diesem fügen wir ein neues Element des Typs **ADO.NET Entity Data Model** namens **DataAnnotationContext** hinzu. Im Dialog **Modellinhalt auswählen** selektieren wir **Leeres Code First-Modell**. Dies fügt die Klasse **DataAnnotationContext.vb**, einige Verweise und Informationen über das Entity Framework und die Verbindungszeichenfolge zur Datei **App.config** hinzu. Normalerweise beziehen sich diese auf eine Datenbank, die den gleichen Namen wie das Projekt plus einem Punkt plus dem Namen der Context-Klasse hat und die mit der aktuellen SQL Server LocalDb-Instanz verwaltet wird.

Entitätsklassen hinzufügen

Zu Beispielzwecken benötigen wir die gleichen Klassen wie im oben genannten Artikel. Diese heißen **Person.vb** und **Anrede.vb** sowie **Fahrzeug.vb** und **Ausstattungsmerkmal.vb**. Wie diese aussehen, entnehmen Sie dem Beispielprojekt, eine Beschreibung finden Sie im oben genannten Artikel. Wichtig ist: Zwischen **Person.vb** und **Anrede.vb** gibt es eine 1:n-Beziehung, zwischen **Fahrzeug.vb** und **Ausstattungsmerkmal.vb** eine m:n-Beziehung.

DbSet-Elemente hinzufügen

Damit der Aufruf der Anweisung **Add-Migration** überhaupt Anweisungen zum Erstellen von Tabellen in eine neue Klasse schreibt, benötigt sie Informationen darüber, welche Tabellen erstellt werden sollen. Diese Information hinterlegen wir in Form von **DbSet**-Deklarationen in der Klasse **DataAnnotationContext.vb**:

```
Public Overridable Property Anreden As DbSet(Of Anrede)
Public Overridable Property Personen As DbSet(Of Person)
Public Overridable Property Fahrzeuge As DbSet(Of Fahrzeug)
Public Overridable Property Ausstattungsmerkmale As DbSet(Of Ausstattungsmerkmal)
```

Erste Migration

Damit können wir die erste Migration durchführen und rufen die Befehle **Enable-Migrations** (erstellt den Ordner **Migrations** mit der Datei **Configuration.vb**) und **Add-Migration init** auf. Anschließend erstellen wir mit **Update-Database** eine Datenbank auf Basis der in der soeben erstellten Datei, deren Dateiname auf **_init.vb** endet. Wenn Sie mit dem Menübefehl **Ansicht|SQL Server-Objekt-Explorer** den Bereich **SQL Server-Objekt-Explorer** aktivieren, können Sie hier eine Verbindung mit der in der Verbindungszeichenfolge angegebenen SQL Server-Instanz herstellen und sich die Datenbank mit ihren Tabellen anzeigen lassen. Diese zeigt nicht nur die vier Tabellen für die von uns hinzugefügten Entitätsklassen an, sondern auch noch eine Verknüpfungstabelle zum Herstellen der Beziehung zwischen den beiden Klassen **Fahrzeug** und **Ausstattungsmerkmal**.

Datenbank neu erstellen

Wenn Sie die Datenbank neu erstellen wollen, sind vor dem Migrieren zwei Schritte nötig:

- Löschen der Datenbank

- Löschen der Dateien im Verzeichnis **Migrations** mit Ausnahme von **Configuration.vb**

Konventionen für das Erstellen/Mapping vom Entity Data Model zum Datenmodell

Entity Framework nutzt hier einige Konventionen, um das Entity Data Model in ein Datenmodell umzusetzen. Diese lauten:

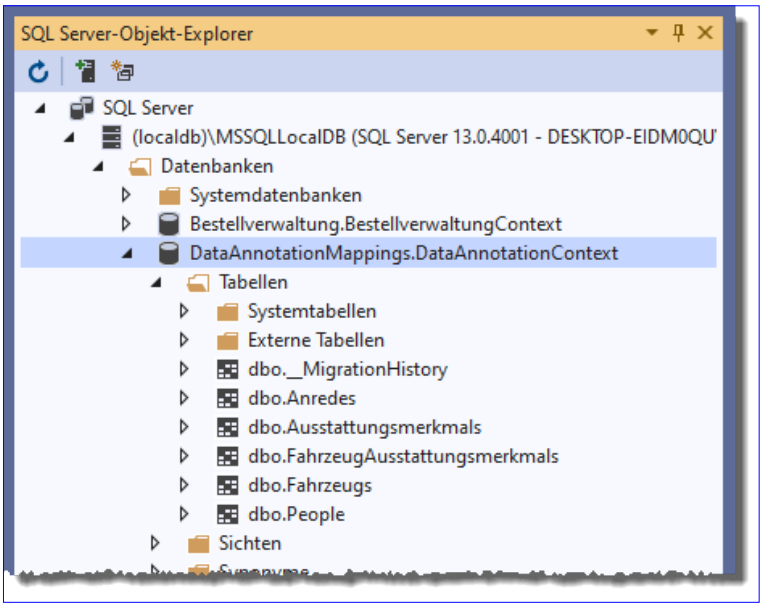


Bild 1: Neue Datenbank mit fünf Tabellen

- Entity Framework erstellt die Datenbank standardmäßig in das Schema **dbo**.

- Tabellennamen leitet Entity Framework aus dem Namen der Entitätsklasse ab und hängt standardmäßig noch ein **s** an. Das ist bei Verwendung englischer Klassennamen meist funktional, weil so aus **Customer** der Tabellenname **Customers** wird. Bei einer deutschen Bezeichnung wie **Kunde** wird **Kundes** daraus, was normalerweise nicht gewünscht sein dürfte. Interessant: Aus der Klasse **Person** erstellt Entity Framework eine Tabelle namens **People**.

- Primärschlüssel wählt Entity Framework nach der Feldbezeichnung aus. Ist ein Feld namens **ID** vorhanden oder ein Feld, dessen Name sich aus der Klassenbezeichnung plus **ID** zusammensetzt, verwendet Entity Framework dieses als Primärschlüsselfeld.

- Fremdschlüsselfelder ermittelt Entity Framework, indem es die Primärschlüsseleigenschaft für die verknüpfte Entität ermittelt, also zum Beispiel **AnredeID**, und dann in der Klasse mit der Fremdschlüsseleigenschaft nach einer gleichnamigen Eigenschaft sucht. Findet es hier eine Eigenschaft namens **AnredeID**, nutzt es diese zum Erstellen des Fremdschlüsselfeldes. Falls nicht, erstellt es ein neues Fremdschlüsselfeld, dessen Bezeichnung sich aus dem Namen der zu verknüpfenden Entität, einem Unterstrich und **ID** zusammensetzt, hier also **Anrede_ID**.

- NULL-Werte legt Entity Framework für Eigenschaften des Typs **String** an. Zahlen- und Datumsfelder werden im Tabellenentwurf so angelegt, dass sie nicht NULL sein dürfen.

- Die Reihenfolge der Felder in den erstellten Tabellen entspricht der Reihenfolge der Eigenschaften der Entitätsklassen. Allein die Primärschlüsselfelder landen ganz oben.

- Entity Framework sieht automatisch die Löschweitergabe für alle Beziehungen vor, wenn das Fremdschlüsselfeld als **Required** markiert ist.

Mapping der Datentypen

Die einzelnen Datentypen der Eigenschaften der Entitätsklassen mappt Entity Framework standardmäßig wie folgt:

- **Integer**: **int**
- **String**: **nvarchar(max)**
- **Decimal**: **decimal(18,2)**
- **Single**: **real**
- **Byte()**: **varbinary(max)**
- **DateTime**: **datetime**
- **Boolean**: **bit**
- **Byte**: **tinyint**
- **Short**: **smallint**
- **Long**: **bigint**
- **Double**: **float**

Konventionen mit DataAnnotations übergehen

Nun schauen wir uns an, wie wir die teilweise nicht erwünschten und per Konvention bewirkten Anpassungen des Datenmodells an das Entity Data Model beeinflussen können. Dazu wollen wir die gleichen Änderungen erreichen wie im Artikel **Code First Mapping per Fluent API (www.datenbankentwickler.net/252)** beschrieben:

- Die Tabellennamen sollen aus dem Präfix **tbl** und dem Entitätsnamen im Plural bestehen, also beispielsweise **tblAnreden**.
- Die Primärschlüsselfelder sollen aus dem Namen der Entität und der Zeichenfolge **ID** bestehen, also beispielsweise **AnredeID**.
- Die Eigenschaft **Name** der Entität **Anrede** soll **Anrede** heißen und den Datentyp **nvarchar(50)** erhalten.
- Die Klasse **Person** enthält als Primärschlüsseleigenschaft das Feld **PK**. Dieses wird nicht per Konvention erkannt und soll als Primärschlüsselfeld mit dem Namen **PersonID** markiert werden.

- Die Klasse **Person** soll natürlich nicht in eine Tabelle namens **People** gemappt werden, sondern in **tblPersonen**.
- Für die Tabelle **tblAnreden** wollen wir das Feld **AnredeID** nicht als Autowert definieren.
- Die Eigenschaften **Vorname** und **Nachname** der Entität **Person** sollen in der Tabelle keine Nullwerte erlauben.
- Die Eigenschaft **Strasse** der Entität **Person** soll nicht in den Tabellenentwurf der Tabelle **tblPersonen** übernommen werden.
- Die Eigenschaft **PLZ** soll den Datentyp **varchar** und die Feldgröße **25** erhalten.
- Die Eigenschaft **AnredeFK** der Entität **Person** soll als Fremdschlüsselfeld zum Herstellen einer Beziehung zur Tabelle **tblAnreden** verwendet werden.
- Die Verknüpfungstabelle zum Herstellen einer Beziehung zwischen den beiden Tabellen **tblFahrzeuge** und **tblAusstattungsmerkmale** soll so angepasst werden, dass die Fremdschlüsselfelder die Bezeichnungen **FahrzeugID** und **AusstattungsmerkmalID** erhalten. Außerdem soll die Tabelle die Bezeichnung **tblFahrzeugeAusstattungsmerkmale** erhalten.
- Wir wollen die Löschweitergabe für die Beziehung zwischen den beiden Tabellen **tblPersonen** und **tblAnreden** deaktivieren.

Um es vorwegzunehmen: Wir werden nicht alle Vorgaben vollständig mit DataAnnotations erledigen können.

Namespaces für DataAnnotations hinzufügen

Die **DataAnnotation**-Elemente befinden sich in eigenen Namespaces, die wir wie folgt zu den einzelnen Entitätsklassen und zur Klasse **DataAnnotationContext.vb** hinzufügen:

```
Imports System.ComponentModel.DataAnnotations
Imports System.ComponentModel.DataAnnotations.Schema
```

Tabellennamen anpassen

Die Anpassung eines Tabellennamens nehmen Sie mit einem **DataAnnotation**-Element vor, dass sie unmittelbar dem Klassennamen der Entitätsklasse zuweisen. Für die Klasse **Person**, die Entity Framework in eine Tabelle namens **tblPersonen** mappen soll, sieht das wie folgt aus:

```
<Table("tblPersonen")>
Public Class Person
    ...
End Class
```

SQL Server Reporting Services

Wer von Access kommt, kennt die dort verfügbare Darstellung von Daten in Berichtsform. Berichte lassen sich mit wenigen Kenntnissen zusammenklicken und werden dann Teil der Datenbankdatei. Wenn Sie in Mehrbenutzerumgebungen Daten als Bericht anzeigen wollen und dabei regelmäßig neue Berichte erstellen, wird das recht aufwändig – neue Berichte müssen dann immer in Form eines neuen Frontends verteilt werden. Oder Sie erstellen ein eigenes Frontend nur für die Berichte. Wie auch immer: Wenn Sie .NET-Anwendungen programmieren, stehen die Access-Berichte nicht mehr bereit. Eine Alternative lautet SQL Server Reporting Services. Diese bieten einen anderen Ansatz als die Access-Berichte: Sie stellen ihre Daten über eine Service bereit, den Sie über den Browser nutzen können. Der Benutzer benötigt also noch nicht einmal Zugriff auf eine Datenbankanwendung, um Berichtsdaten einzusehen. Der vorliegende Artikel liefert grundlegende Informationen über die SQL Server Reporting Services.

Die SQL Server Reporting Services

Bei den SQL Server Reporting Services handelt es sich um ein eigenständiges Produkt, dass aber eng mit den Programmiertools verknüpft ist. Seine Aufgabe ist es nicht nur, wie Access Berichte auf Basis der in den Tabellen gespeicherten Daten nach verschiedenen Kriterien aufzubereiten und auszugeben.

Es ist eher eine komplette Infrastruktur zur Erstellung, Verwaltung und Bereitstellung von Berichten in einem Unternehmen. Damit erhalten Sie mit den SQL Server Reporting Services einen Baustein für den Bereich der Business Intelligence. Damit soll es möglich sein, die für verschiedene Bereiche des Unternehmens notwendigen Daten zu liefern – ob übersichtliche Dashboards für die Geschäftsführung, detaillierte Daten für das Controlling oder einfach Artikellisten mit Verfügbarkeiten und Lieferzeiten.

Installation und Co.

Wie Sie die SQL Server Reporting Services, kurz SSRS, installieren, welche weiteren Elemente Sie dafür benötigen und wie Sie erste Beispielberichte damit erstellen und veröffentlichen, zeigen wir beispielsweise in den Artikeln **Reporting Services 2019 installieren und starten (www.datenbankentwickler.net/256)** und **Reporting Services: Web-Portal (www.datenbankentwickler.net/257)**.

Der vorliegende Artikel soll grundlegende Informationen zu den SQL Server Reporting Services liefern.

Was können die SQL Server Reporting Services?

Mit den SSRS können Sie Berichte erzeugen, bereitstellen und verwalten. Dabei erfolgt der Zugriff durch den Anwender (nicht zu verwechseln mit dem Programmierer der Berichte) über eine Webschnittstelle. Mit den SSRS können Sie Berichte in verschiedenen Formaten erzeugen, zum Beispiel:

- Paginierte Berichte in Form von PDF-Dateien, Excel-Dateien, Word-Dateien et cetera

- Mobile Berichte, die der Benutzer auf Smartphones oder Tablet lesen kann. Diese Berichte werden speziell darauf ausgelegt, auf unterschiedlichen Bildschirmgrößen dargestellt zu werden.

- Webbasierte Berichte für die Anzeige im Webbrowser

Berichte enthalten Elemente wie Texte, Tabellen, Grafiken, Bilder oder Diagramme.

Berichte definieren mit der Report Definition Language (RDL)

Die Sprache für die Definition von Berichten heißt **Report Definition Language**, kurz **RDL**. Dabei handelt es sich um eine XML-basierte Sprache. Ein RDL-Dokument enthält die kompletten Informationen, die zum Erstellen eines Berichts notwendig sind. Diese teilen sich auf in Informationen für das Ermitteln der Daten und das Layout, in dem die Daten dargestellt werden sollen. Durch das XML-Format ist auch ein Austausch mit anderen Anwendungen möglich.

Die RDL-Datei als das Ergebnis des Designs eines Berichts enthält somit Vorgaben dafür, wie die aus den angegebenen Datenquellen ermittelten Daten angeordnet werden sollen. Details über die Sprache RDL benötigen Sie in der Regel nicht, denn es gibt verschiedene Tools, mit denen Sie Berichte per Drag and Drop über eine entsprechende Benutzeroberfläche zusammenstellen können.

Microsoft SQL Server Services für Business Intelligence

Die SQL Server Reporting Services sind nicht allein, wenn es darum geht, Business Intelligence-Auswertungen bereitzustellen. Es gibt noch zwei weitere Services neben den SSRS, die für den Datenzugriff und die Auswertung verwendet werden können. Insgesamt haben wir also:

- **SQL Server Reporting Services** (**SSRS**)

- **SQL Server Analysis Services** (**SSAS**)

- **SQL Server Integration Services** (**SSID**)

SQL Server Analysis Services (SSAS)

Die SQL Server Analysis Services, kurz SSAS, dienen der Analyse umfangreicher Datenmengen mit verschiedenen Methoden und Verfahren. Damit erhalten Sie ein Tool, das multidimensionales Online Analytical Processing und Data Mining im Business Intelligence-Bereich erlaubt. Die zu analysierenden Daten können aus verschiedenen Datenquellen wie relationalen Datenbanken, XML-Dokumenten oder auch Textdateien stammen und werden mit den nachfolgend vorgestellten SQL Server Integration Services erfasst. Die SSAS erlauben ein besseres Verständnis von Geschäftsdaten eines Unternehmens.

SQL Server Integration Services (SSIS)

Die SQL Server Integration Services, kurz SSIS, haben die Aufgabe, Daten aus unterschiedlichen Datenquellen zusammenzuführen. Mit den SQL Server Integration Services fassen Sie Daten aus verschiedenen Quellen

Reporting Services 2019 installieren und starten

Der Vorteil von Microsoft Access ist: Man bekommt alles aus einer Hand. Tabellenentwurf, Abfragedesigner, Formulare, Programmierumgebung und – Tools zur Berichterstellung. Bei der Datenbank mit Visual Studio gibt es so viele Möglichkeiten, die aber alle irgendwie viel größer und komplizierter erscheinen als das, was der Berichtsdesigner von Access bietet. Früher oder später wollen wir aber auch in DATENBANKENTWICKLER das Thema Reporting behandeln und deshalb schauen wir uns in dieser Artikelreihe die Möglichkeiten der SQL Server Reporting Services an. Da SQL Server mittlerweile in einer Community Version kommt, die für den privaten Einsatz kostenlos ist, steht dem Ausprobieren auch nichts im Wege. Der vorliegende Artikel ist eine Neuauflage des Artikels »Reporting Services: Installation und Start« (www.datenbankentwickler.net/200). Diese ist notwendig, weil sich einige Schritte beim Installieren geändert haben.

Schritt für Schritt

In den folgenden Abschnitten zeigen wir Ihnen, welche Tools und Software Sie benötigen, um die Reporting Services für den SQL Server in der Version von 2019 verwenden zu können. Dabei erledigen wir die folgenden Schritte:

- Installation der **SQL Server 2019 Reporting Services**
- SQL Server-Instanz für die Verwendung von **SQL Server 2019 Reporting Services** festlegen
- **SQL Server Data Tools** installieren
- **SQL Server Integration Services Projects** installieren
- **Microsoft Analysis Services Projects** installieren
- **Microsoft Reporting Services Projects** installieren

Bild 1: Installation von **SQL Server 2019 Reporting Services**

Installation der Reporting Services

Die SQL Server Reporting Services finden Sie, wenn Sie bei Google nach den Begriffen **Installieren von SQL Server Reporting Services** suchen. Auf der Seite von Microsoft mit diesem Titel finden Sie einen Downloadlink für **SQL Server 2019 Reporting Services**.

Die dort heruntergeladene Datei **SQLServerReportingServices.exe** starten Sie anschließend, um die **SQL Server 2019 Reporting Services** zu installieren. Hier finden Sie zunächst nur die Option aus Bild 1 vor.

Im folgenden Schritt wählen Sie dann aus, welche Edition Sie installieren wollen. Wenn Sie eine gekaufte Lizenz haben, verwenden Sie natürlich den entsprechenden **Product Key**.

Anderenfalls erhalten Sie mit der Entwickler-Edition, die wir in Bild 2 ausgewählt haben, den vollen Funktionsumfang. Ein Klick auf **Weiter** zeigt die Lizenzvereinbarungen an.

Danach folgen die Abfrage, ob Sie bereits einen SQL Server installiert haben, was Sie gegebenenfalls nun nachholen können. Danach wählen Sie den Installationsspeicherort aus, wo Sie in der Regel das vorgegebene Verzeichnis übernehmen können, und starten die Installation mit einem Klick auf die Schaltfläche **Installieren** (siehe Bild 3).

Nach wenigen Augenblicken hat der Installationsassistent **SQL Server 2019 Reporting Services** installiert und zeigt den Abschlussbildschirm an (siehe Bild 4).

Hier haben Sie nun die Möglichkeit, den Berichtsserver manuell zu konfigurieren. Dazu klicken Sie auf die Schaltfläche **Berichtsserver konfigurieren**.

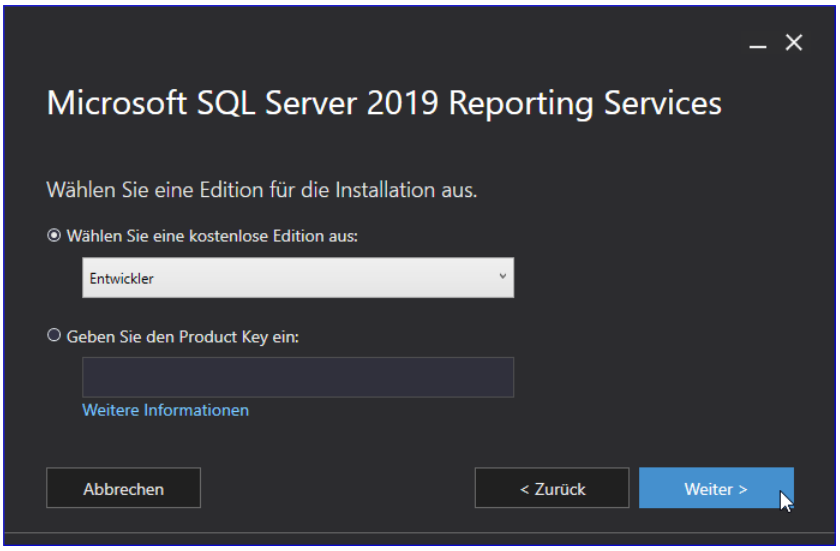


Bild 2: Zweiter Schritt des Assistenten

Bild 3: Start der Installation

Bild 4: Abschluss und weitere Optionen

SQL Server-Instanz auswählen

Dies startet den **Berichtsserver-Konfigurations-Manager**. Dieser verlangt zunächst nach der Auswahl des zu verwendenden SQL Servers. Hier wollen wir, da wir den Berichtsserver für die Beispieldatenbanken einer **LocalDb**-Instanz nutzen wollen, diese Instanz wählen. Allerdings gelingt dies nicht: **LocalDb** ist eine Instanz, die in der Desktop Session läuft, und daher nicht für den Einsatz mit den **SQL Server 2019 Reporting Services** zur Verfügung steht. Daher geben Sie eine als Service laufende SQL Server-Instanz an und bestätigen den Dialog mit **Verbinden** (siehe Bild 5).

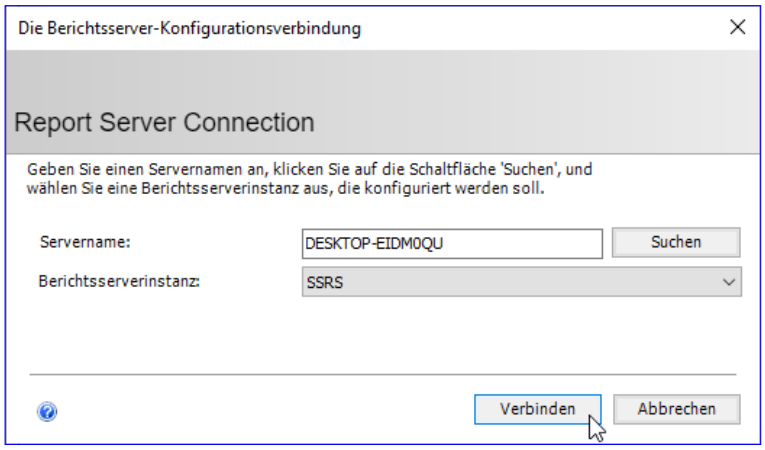


Bild 5: Einstellen der Verbindung für den Report Server

Report Server nachträglich anpassen

Wenn Sie diesen Schritt auslassen oder die Konfiguration nachträglich ändern wollen, können Sie den Berichtsserver-Konfigurations-Manager auch später noch starten. Dazu geben Sie im Suchfeld von Windows den Suchbegriff **Report Server Configuration Manager** ein und starten diesen Eintrag anschließend. Die Einstellungen zeigt der Berichtsserver-Konfigurations-Manager anschließend in einer Übersichtsseite an, die Sie auch erhalten, wenn Sie diesen erneut starten (siehe Bild 6).

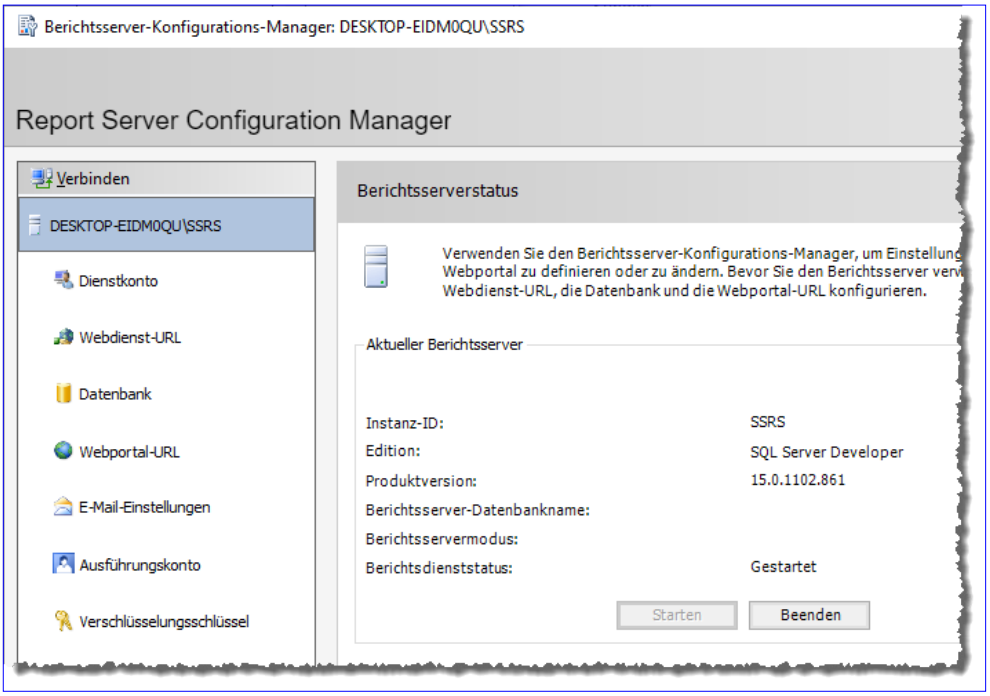


Bild 6: Der Berichtsserver-Konfigurations-Manager

Report Server-Datenbank anlegen

Hier sind noch weitere Einstellungen nötig. Zum Beispiel legen wir eine Datenbank an, welche die Reporting Services nutzen können. Dazu wählen Sie im linken Bereich des **Report Server Configuration Manager** den Eintrag **Datenbank** aus. Anschließend klicken Sie im nun erscheinenden Bereich **Berichtsserver-Datenbank** auf die Schaltfläche **Datenbank ändern** (siehe Bild 7).

Bild 7: Einstellen einer Datenbank für die Reporting Services

Im nun erscheinenden Dialog **Assistent zum Konfigurieren der Berichtsserver-Datenbank** behalten Sie die Option **Neue Berichtsserver-Datenbank erstellen** bei und klicken auf die Schaltfläche **Weiter** (siehe Bild 8).

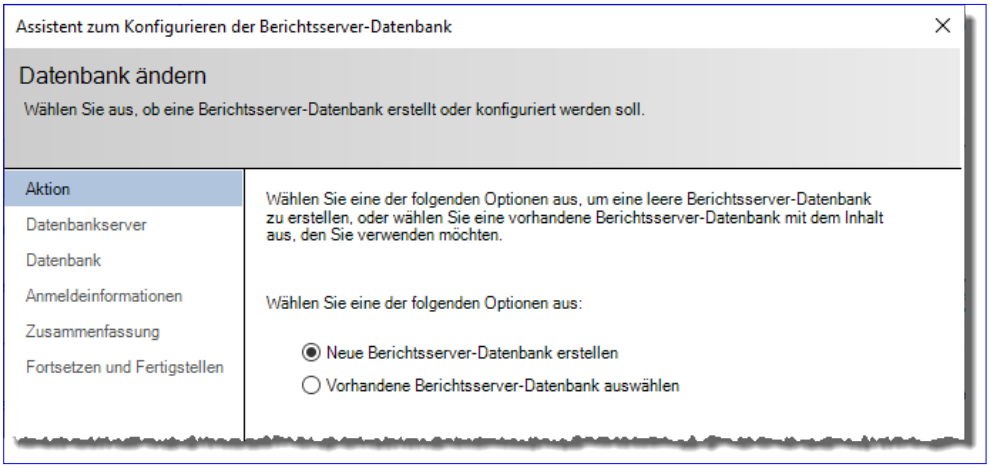


Bild 8: Erstellen einer neuen Datenbank für die Reporting Services

Im nächsten Schritt namens **Datenbankserver** behalten Sie die aktuellen SQL Server-Instanz bei oder wählen eine andere Instanz aus. Der folgende Schritt mit der Bezeichnung **Datenbank** erwartet die Angabe eines Namens für die zu erstellende Datenbank. Hier behalten wir

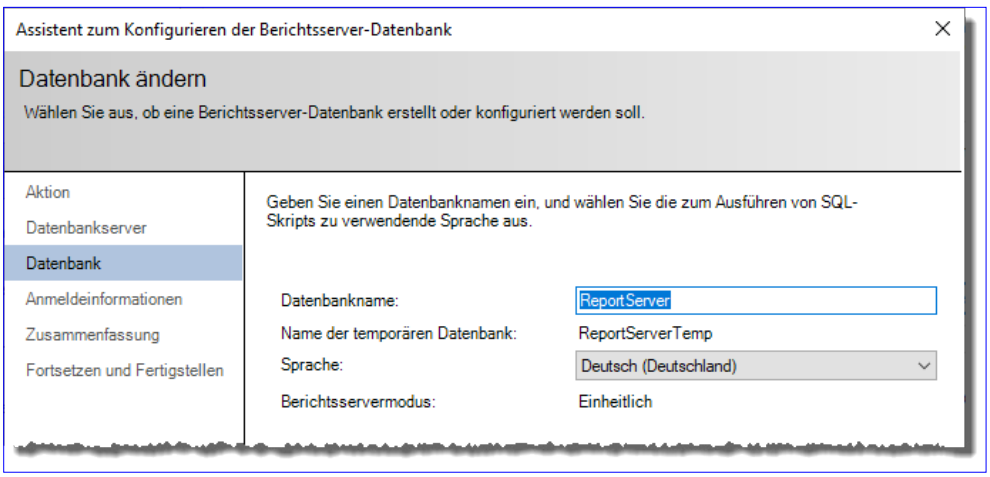


Bild 9: Anlegen der neuen Datenbank für die Reporting Services

Design festlegen

Der nächste Schritt bietet die Möglichkeit, die Felder der als Datenquelle festgelegten Tabelle auf die verschiedenen Bereiche des Berichts wie **Seite**, **Gruppieren** und **Details** aufzuteilen. Zu Beispielzwecken fügen wir einfach alle Element zum Bereich **Details** hinzu.

Im letzten Schritt erhalten wir noch eine Übersicht und können die Erstellung des Berichts dann mit einem Klick auf die Schaltfläche **Fertig stellen** abschließen.

Der Bericht erscheint dann in der Entwurfsansicht in Visual Studio (siehe Bild 21).

Mit einem Klick auf den Reiter **Preview** zeigen Sie die Berichtsvorschau an (siehe Bild 22).

Zusammenfassung und Ausblick

Damit haben wir den ersten Bericht mit Visual Studio 2019 und SQL Server 2019 erstellt.

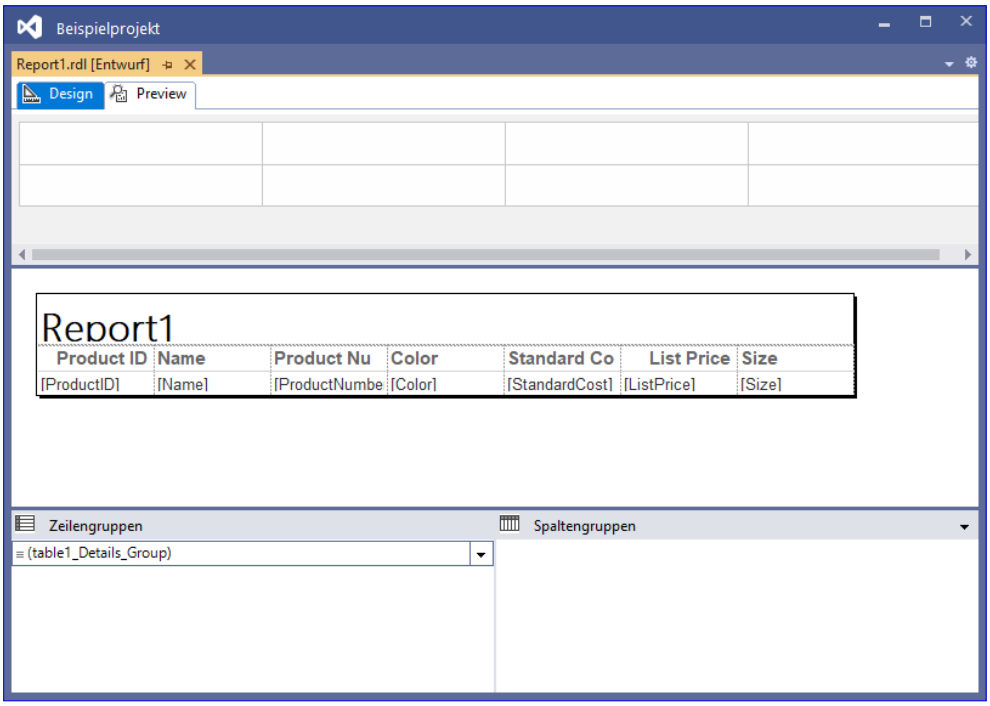


Bild 21: Entwurf des Berichts in Visual Studio

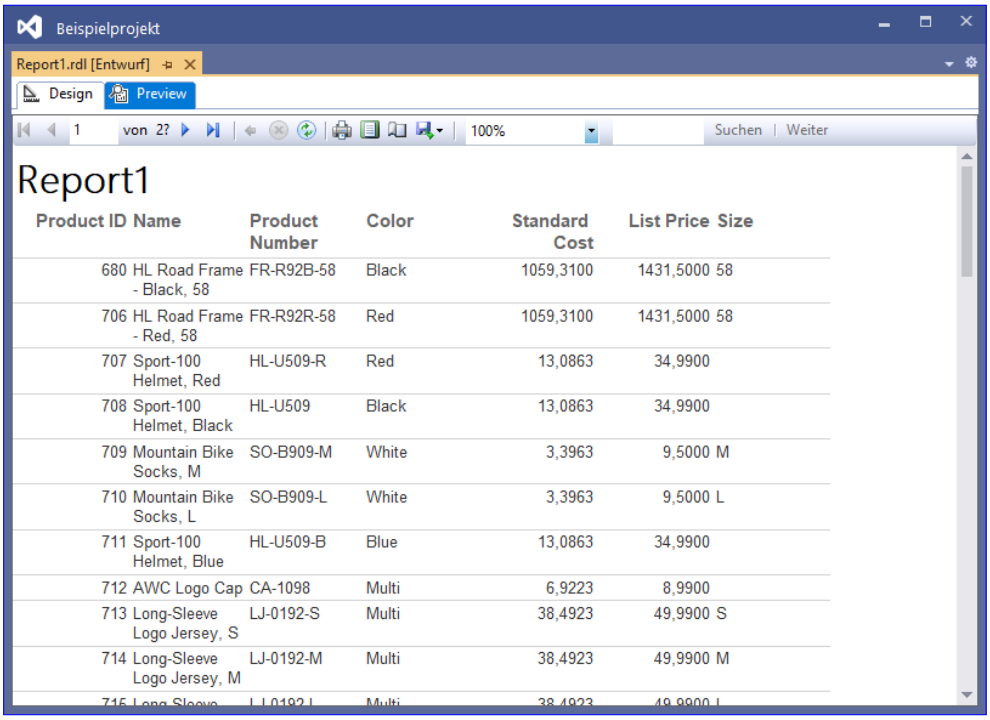


Bild 22: Vorschau des Berichts mit Livedaten

In einem weiteren Artikel namens **Reporting Services: Web-Portal** (**www.datenbankentwickler.net/257**) zeigen wir, wie Sie den Bericht für die Benutzer verfügbar machen.

SQL Server Report Builder

Neben Visual Studio gibt es noch ein weiteres, wesentlich schlankeres Tool zum Definieren von Berichten auf Basis der SQL Server Reporting Services. Dieses Tool heißt Report Builder und kann kostenlos bei Microsoft heruntergeladen werden. Es bietet sich als Alternative für solche Anwendungszwecke an, wo nur Berichte designed werden sollen und die übrigen Funktionen von Visual Studio nicht erforderlich sind. Dieser Artikel zeigt, wie Sie den Report Builder installieren und verwenden.

Download des Report Builders

Den Download des Report Builders finden Sie, wenn Sie in einer Suchmaschine den Begriff **SQL Server Report Builder** eingeben.

Installation des Report Builders

Die Installation erfolgt auf einfache Weise durch Starten der zuvor heruntergeladenen **.msi**-Datei. Der einzige Schritt, in dem Sie etwas erledigen können, ist der aus Bild 1.

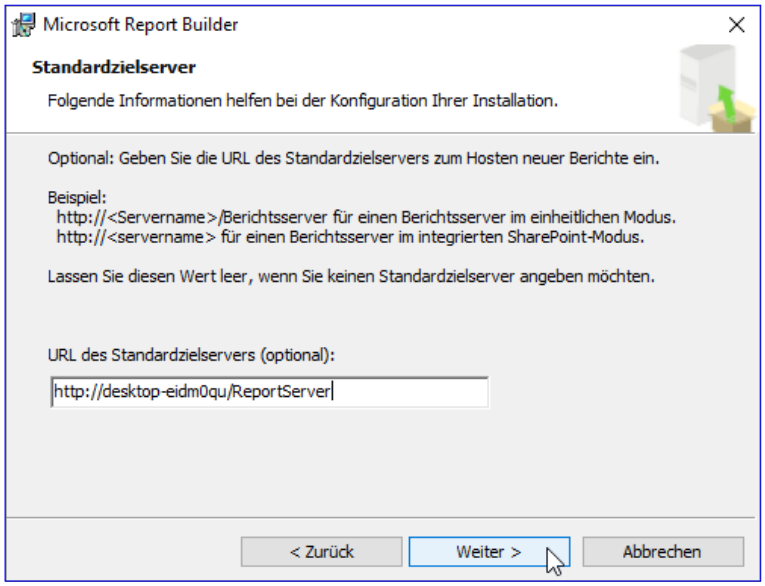


Bild 1: Angabe des Berichtsservers während der Installation

Für die Eigenschaft **URL des Standardzielservers** geben Sie die im Report Server Configuration Manager angegebene URL unter **Webdienst-URL** an. Den Report Server Configuration Manager öffnen Sie über das **Suchen**-Textfeld von Windows.

Im **Report Server Configuration Manager** wechseln Sie zum Bereich **Webdienst-URL** und ermitteln dort unter URLs für Berichtsserver-Webdienst die gesuchte URL (siehe Bild 2).

In unserem Fall lautet die hier einzutragende URL:

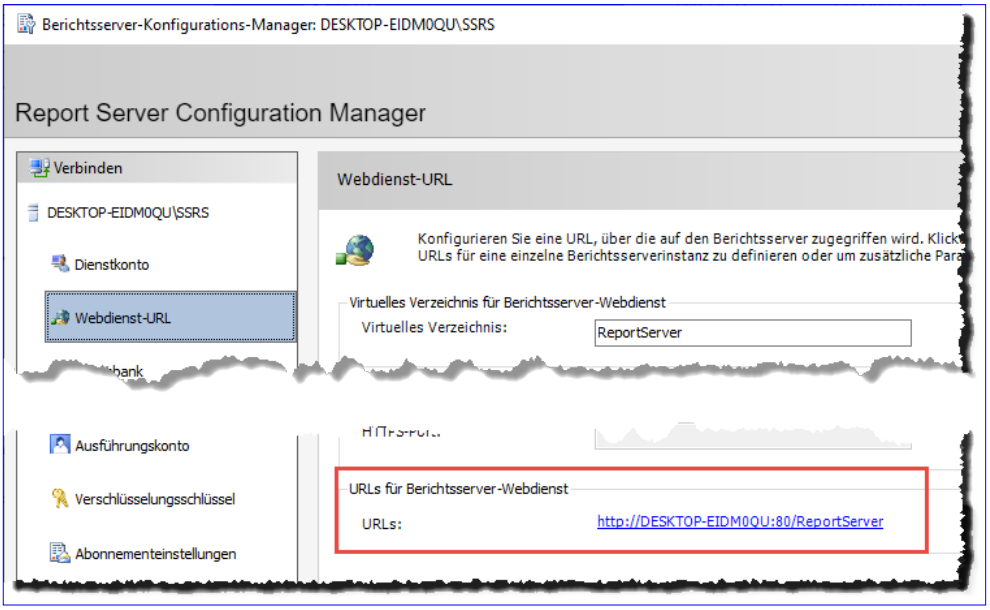


Bild 2: Ermitteln der im Report Server Configuration Manager definierten Webdienst-URL

```
http://desktop-eidm0qu/ReportServer
```

Wenn die Reporting Services auf dem lokalen Rechner betrieben werden, können Sie auch **localhost** als Server angeben.

Danach starten Sie die Installation, die innerhalb weniger Sekunden abgeschlossen ist. Den Report Builder können Sie anschließend über das Suchen-Fenster von Windows finden und starten.

Report Builder vom Web-Portal aus starten

Eine weitere Möglichkeit, den Report Builder zu starten, ist das Web-Portal. Dieses finden Sie im Webbrowser unter **<Servername>/Reports**, für Reporting Services auf dem lokalen Rechner also beispielsweise **localhost/Reports**. Wenn Sie hier auf das Plus-Zeichen klicken, erscheint eine Liste von Optionen zum Anlegen neuer Elemente (siehe Bild 3).

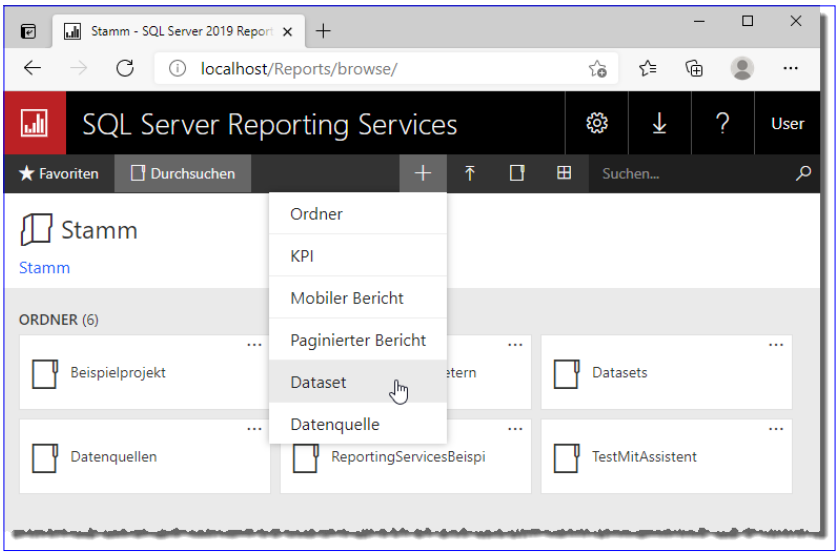


Bild 3: Anlegen neuer Objekte über das Web-Portal

Wenn Sie den Report Builder zuvor noch nicht vom Web-Portal aus geöffnet haben, erscheinen die Meldungen aus Bild 4.

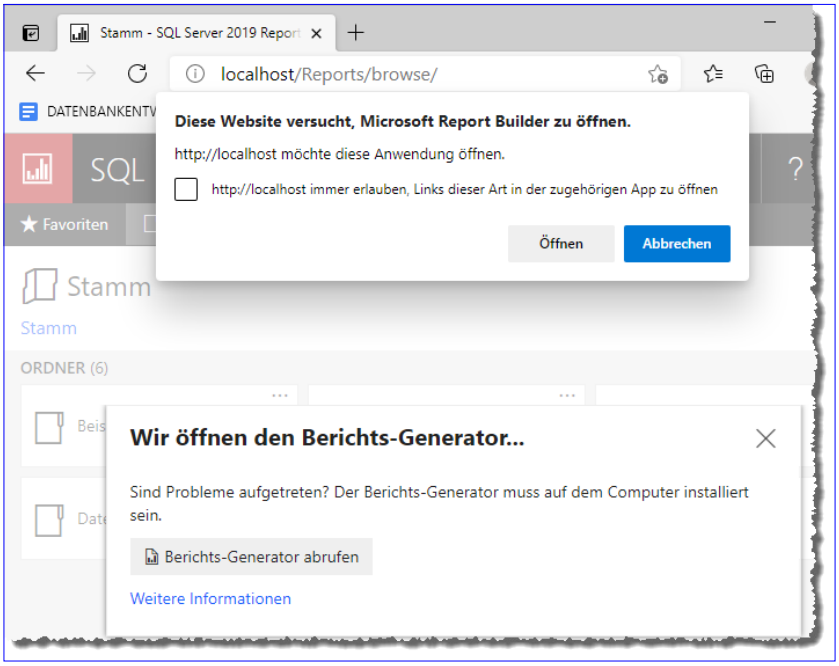


Bild 4: Freigeben des Öffnens des Report Builders

Gegebenenfalls erhalten Sie auch noch die Meldung aus Bild 5, die Sie bestätigen müssen, um die Verbindung herzustellen.

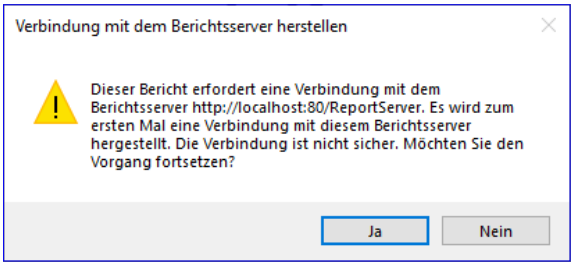


Bild 5: Sicherheitsabfrage des Berichtsservers

Danach wird der Report Builder geöffnet. Je nachdem, welchen Befehl Sie verwendet haben, zeigt der Report Builder in seinem Startbildschirm nur einen Bereich an. Im Folgenden der gewählte Befehl und der angezeigte Bereich im Startbildschirm:

- **Paginierter Bericht**: Öffnet den Report Builder mit dem Bereich **Neuer Bericht**.

- **Neues Dataset**: Öffnet den Report Builder mit dem Bereich **Neues Dataset**.

Erste Schritte im Report Builder

Der Report Builder erwartet sie mit einem aufgeräumten Startfenster. Dieses bietet nach dem Öffnen über Windows vier Bereiche an, nämlich **Neuer Bericht**, **Neues Dataset**, **Öffnen** und **Zuletzt verwendet** (siehe Bild 6) – also mehr Bereiche als beim Öffnen des Report Builders vom Web-Portal aus.

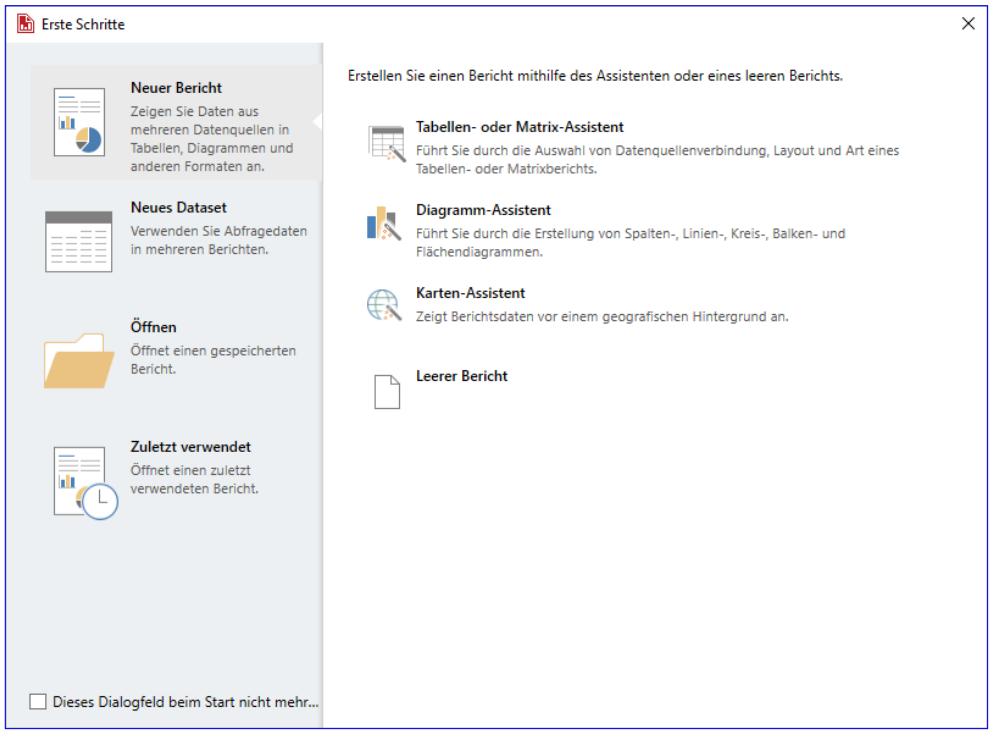


Bild 6: Der Startdialog des Report Builders

Hier finden Sie verschiedene Bereiche vor, die wiederum unterschiedliche Optionen anbieten. Da das Erstellen eines neuen Berichts für die beiden Arten **Tabellen- oder Matrix-Assistent** beziehungsweise **Diagramm-Assistent** das Vorhandensein eines Datasets erwartet, könnten wir uns hier zuerst den Bereich **Neues Dataset** ansehen. Wir können uns aber auch direkt von dem Startdialog verabschieden und die benötigen Elemente direkt über die dafür vorgesehenen Funktionen des Report Builders erstellen.

Dazu schließen Sie einfach das Startfenster und wenden sich dem Bereich Berichtsdaten zu (siehe Bild 7).

Der neue Bericht ist bereits vorhanden und wird in der Entwurfsansicht angezeigt, wir benötigen also als nächstes eine Datenquelle und ein Dataset.

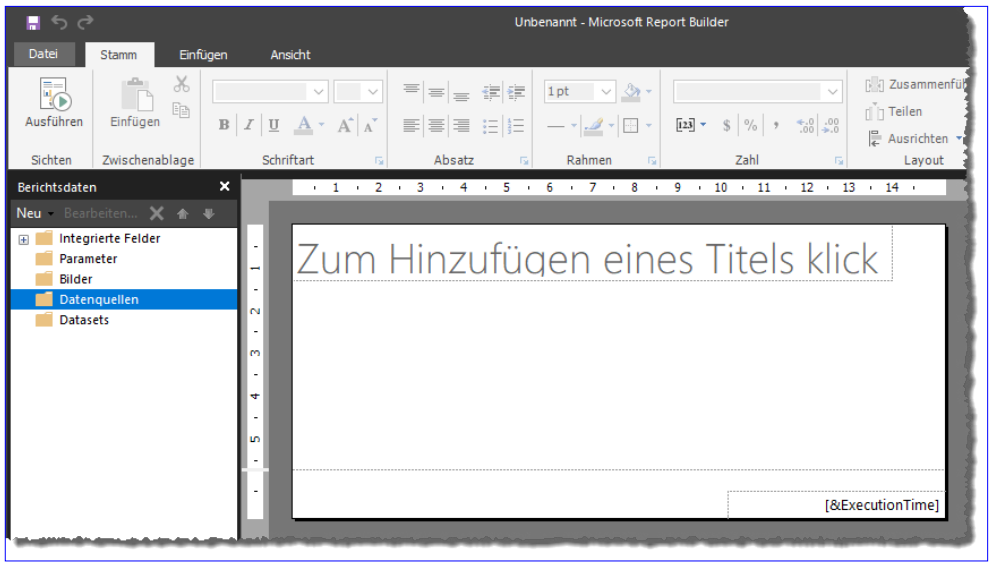


Bild 7: Startpunkt zum Erstellen von Elementen im Report Builder

Reporting Services: Web-Portal

In den bisherigen Artikeln über die Erstellung von Berichten mit den Reporting Services haben wir einen Aspekt noch nicht betrachtet: Wie kann der Benutzer eigentlich auf die Berichte zugreifen? Immerhin benötigen wir für die Definition von Reporting Services immer eigene Projekte, die parallel beispielsweise zu unseren WPF-Anwendungen erstellt werden müssen. Dieser Artikel bringt Licht ins Dunkel und zeigt, wie Sie Berichte im Web-Portal öffnen und diesen auch noch Parameter zum Filtern von Daten übergeben können. Außerdem zeigen wir, wie Sie Berichte von WPF-Anwendungen aus aufrufen.

Veröffentlichung von Berichten vorbereiten

Die Installation der Reporting Services haben wir im Artikel **Reporting Services 2019 installieren und starten (www.datenbankentwickler.net/256)** besprochen.

Dort haben wir noch nicht die Einstellungen vorgenommen, die notwendig sind, damit Sie die von den Reporting Services generierten Berichte auch über das Web-Portal anzeigen oder von anderen Anwendungen aus abrufen können – so, wie Sie es beispielsweise von Access gewohnt sind, wo Sie einfach auf eine Schaltfläche klicken und dann der gewünschte Bericht erzeugt wird.

Um die Einstellungen nachträglich vorzunehmen, rufen Sie über die Suche von Windows die Anwendung **Report Server Configuration Manager** auf. Dieser

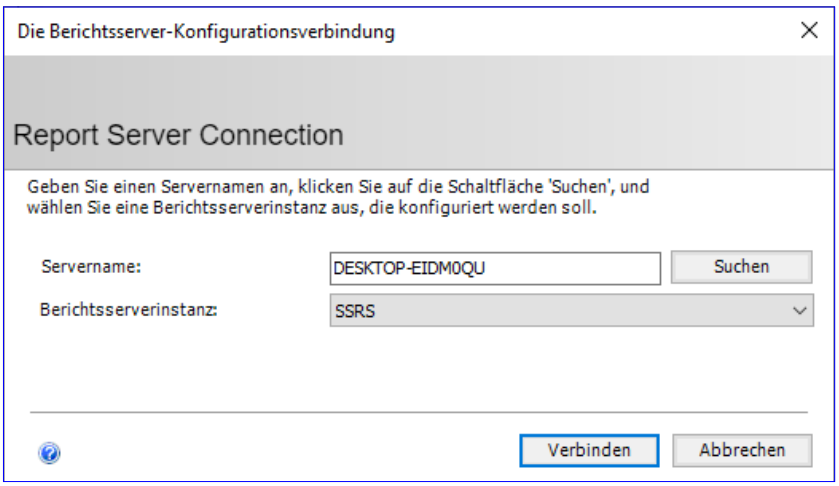


Bild 1: Erstellen einer Verbindung

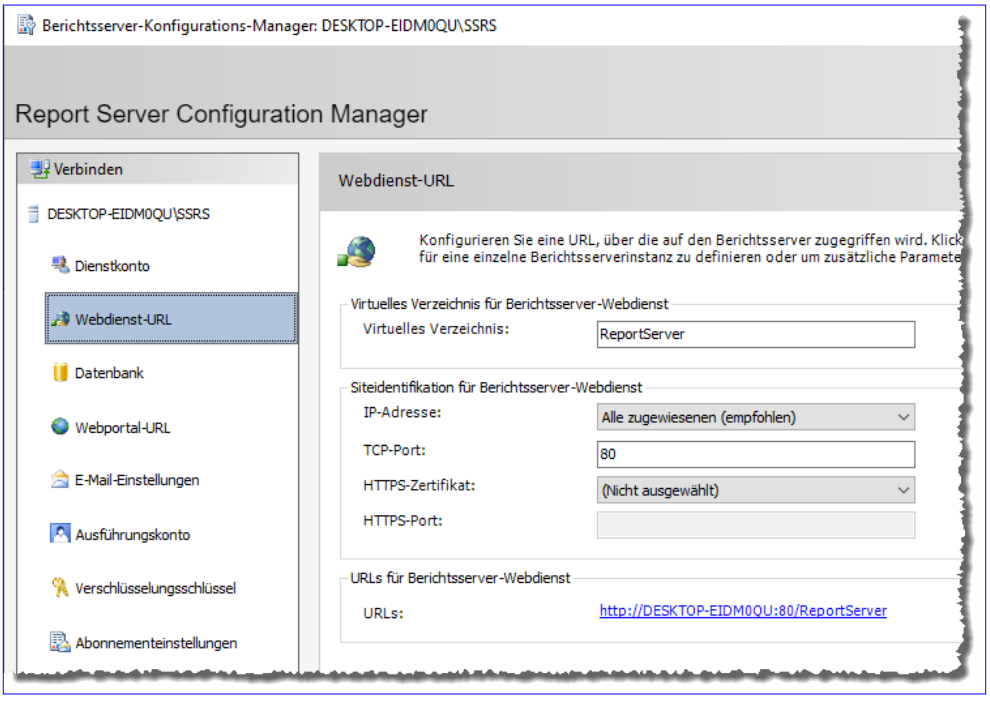


Bild 2: Einstellen der URL für den Webdienst

erwartet Sie mit dem Dialog aus Bild 1, mit dem Sie den Server und die Instanz auswählen.

Webdienst-URL erzeugen

Danach wechseln Sie im Fenster **Report Server Configuration Management** auf die Seite **Webdienst-URL** (siehe Bild 2). Hier sind bereits einige Werte voreingestellt, die Sie einfach durch Betätigen der Schaltfläche **Anwenden** übernehmen können, um den Webdienst mit diesen Werten zu initialisieren. Die wichtigste Information, die wir danach erhalten, ist die URL, die in unserem Fall wie folgt lautet:

```
http://desktop-eidm0qu/ReportServer
```

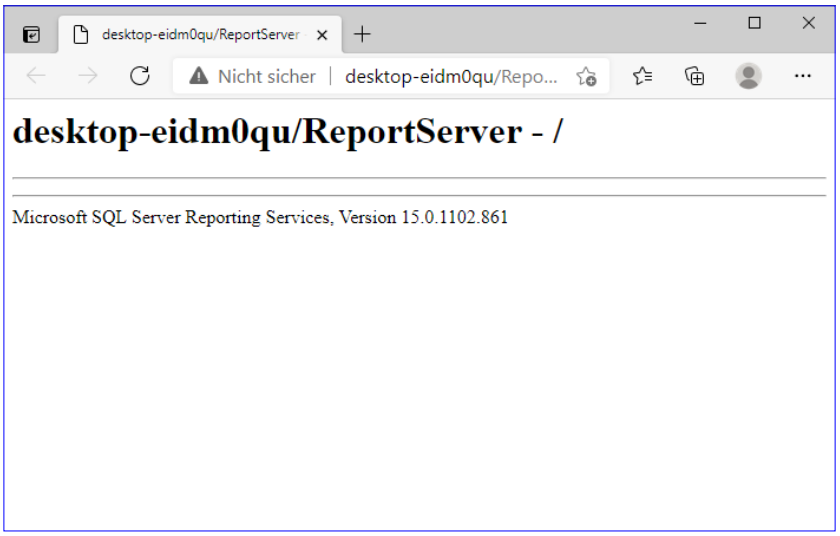


Bild 3: Erster Aufruf des Report-Servers

Die Einstellungen auf den beiden weiteren Seiten **Datenbank** und **Webportal-URL** übernehmen wir ebenfalls durch Betätigen der Schaltfläche **Anwenden**.

Erster Aufruf

Danach rufen wir erstmalig die Seite des Report-Servers auf. Dies liefert die recht leere Webseite aus Bild 3.

Berichte veröffentlichen

Um einen ersten Bericht zu veröffentlichen, öffnen Sie ein **Reporting Services Project**-Projekt – also beispielsweise das aus dem Download zum vorliegenden Artikel.

Dort klicken Sie im Projektmappen-Explorer mit

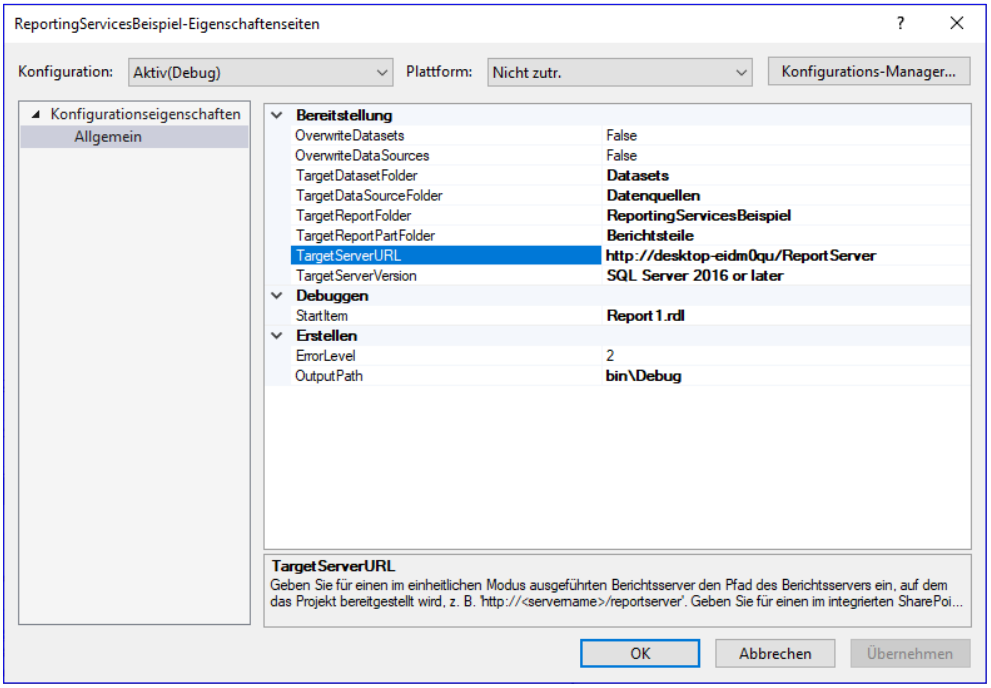


Bild 4: Einstellen der Eigenschaft **TargetServerURL**

der rechten Maustaste auf das **Projekt**-Element und wählen aus dem Kontextmenü den Eintrag **Eigenschaften** aus.

Dadurch öffnet sich der Eigenschaften-Dialog aus Bild 4. Hier finden wir unter anderem die Eigenschaft **TargetServerURL**.

Diese lautet standardmäßig **http://localhost/reportserver**.

Wenn sich der Report Server auf dem lokalen Rechner befindet, können sie diesen Wert beibehalten. Anderenfalls stellen wir den folgenden Wert ein und ersetzen dabei **<Server>** durch den Servernamen:

```
http://<Server>/Report-
Server
```

Nach dem Schließen der Eigenschaften erstellen wir das Projekt, indem wir aus dem Kontextmenü des Projekt-Elements im Projektmappen-Explorer den Eintrag **Erstellen** aufrufen.

Bild 5: Ausgabe zum Bereitstellen des Berichts

Bild 6: Anzeige des Beispielprojekts **ReportingServicesBeispiel**

Bild 7: Anzeige des Beispielberichts **Report1**

In welcher Form wurde das Projekt nun erstellt? Wenn wir uns die Webseite der Reporting Services ansehen, finden wir dort keine neuen Elemente gegenüber dem vorherigen Zustand vor. Warum werden diese dort nicht angezeigt?

Reporting Services Datenquellen

Wenn Sie die Reporting Services nutzen wollen, um Berichte anzuzeigen, benötigen Sie dazu vor allem eines: Die Daten, die in den Berichten abgebildet werden sollen. Sie können nicht einfach wie in einer Access-Datenbank auf die Daten der aktuellen Datenbank zugreifen. Reporting Services sind vielmehr eine von den Datenquellen getrennte Instanz, der Sie erst einmal bekanntgeben müssen, welche Daten verwendet werden sollen. Das erledigen Sie mit der Definition sogenannter Datenquellen. Diese definieren Sie in verschiedenen Anwendungen. Alles rund um die Definition von Datenquellen für Reporting Services erfahren Sie in diesem Artikel!

Welche Reporting Services Datenquellen gibt es?

Wenn Sie mit Reporting Services arbeiten, sind Sie im Gegensatz zu anderen Anwendungen wie Microsoft Access, wo Tabellen und Berichte in einer einzigen Datei untergebracht sind, nicht von einer einzigen Datenquelle abhängig. Streng genommen können Sie natürlich auch in einer Access-Datenbank Daten aus verschiedenen Quellen sammeln, indem Sie entsprechende Verknüpfungen dafür hinterlegen. Mit den Reporting Services haben Sie allerdings eine eigene Instanz, die nicht speziell etwa mit einer bestimmten SQL Server-Datenbank verknüpft ist. Stattdessen können Sie die zu erstellenden Berichte und Darstellungen unabhängig von den Anwendungen, in denen Sie die Daten verwalten, zusammenführen. Dementsprechend gibt es auch separate Möglichkeiten, um diese Berichte abzurufen – in der Regel über das Web-Portal der Reporting Services.

Sie können beispielsweise die folgenden Typen von Datenquellen nutzen:

- Microsoft SQL Server- und Microsoft Azure-Datenbanken
- Oracle, SAP BW, Hyperion, SharePoint-Listen, Teradata
- OLE DB
- ODBC
- XML

Die Übersicht sieht überschaubar aus. Allerdings erhalten Sie allein mit ODBC nochmals Zugriff auf viele weitere Datenformate wie weitere Datenbanktypen, Excel-Tabellen et cetera.

Reporting Services Datenquellen mit verschiedenen Tools definieren

Microsoft bietet verschiedene Tools zum Definieren von Reporting Services Datenquellen an. Dabei handelt es sich um die folgenden – die Möglichkeiten in den verschiedenen Assistenten lassen wie einmal außen vor:

- Visual Studio: in Projekten des Typs **Berichtsserver-Assistent** sowie **Berichtsserverprojekt**

- Microsoft Report Builder
- Web-Portal der Reporting Services

In den folgenden Abschnitten schauen wir uns die verschiedenen Möglichkeiten im Detail an.

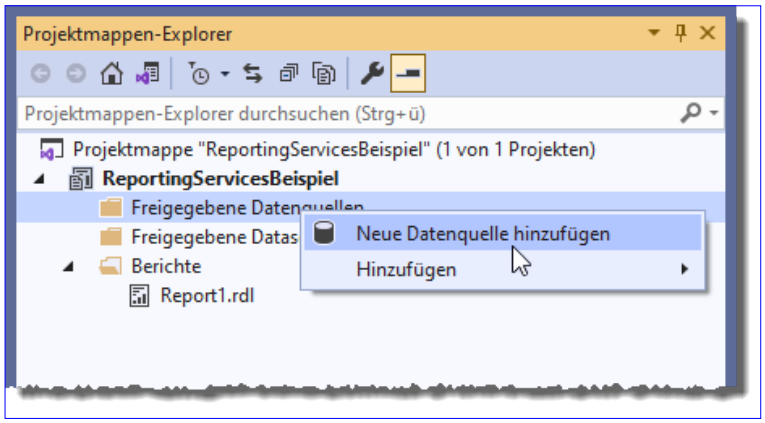


Bild 1: Hinzufügen einer Datenquelle in Visual Studio

Eingebettete oder freigegebene Datenquellen

Es gibt zwei Arten von Datenquellen:

- Bei eingebetteten Datenquellen werden die Informationen mit der Definition des Berichts gespeichert. Sie kann dementsprechend auch nur von diesem Bericht verwendet werden.
- Bei freigegebenene Datenquellen sind die Definitionen der Datenquelle für alle Berichte verfügbar, die sich im gleichen Projekt befinden. Sie können freigegebene Datenquellen allerdings nicht von anderen Projekten aus verwenden.

Weiter unten zeigen wir, wie Sie die beiden Arten definieren können.

Reporting Services Datenquellen mit Visual Studio definieren

In Visual Studio definieren Sie Datenquellen für die Reporting Services in Projekten des Typs **Berichtsserverprojekt-Assistent** oder **Berichtsserver-Projekt**.

Nach dem Durchlaufen des Assistenten oder nach dem Öffnen eines Berichtsserver-Projekts ohne Assistent finden Sie im Projektmappen-Explorer einen Ordner namens **Freigegebene**

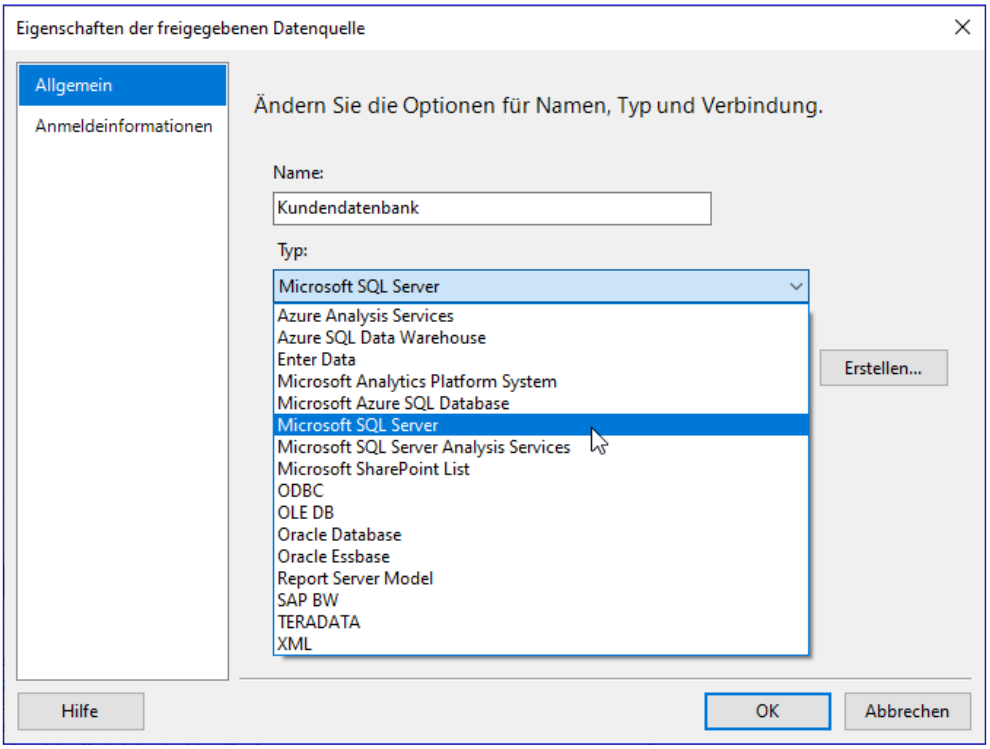


Bild 2: Auswahl des Typs der Datenquelle

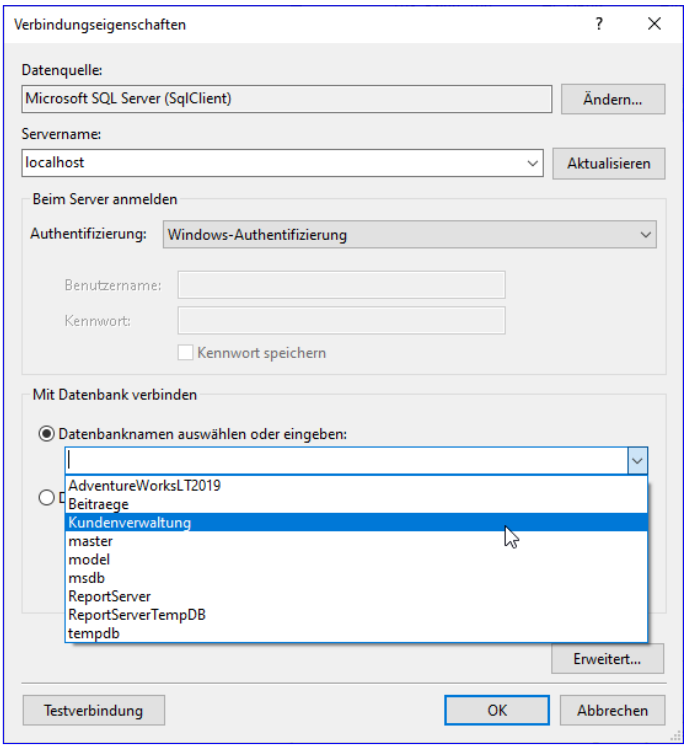


Bild 3: Definieren der Verbindungseigenschaften

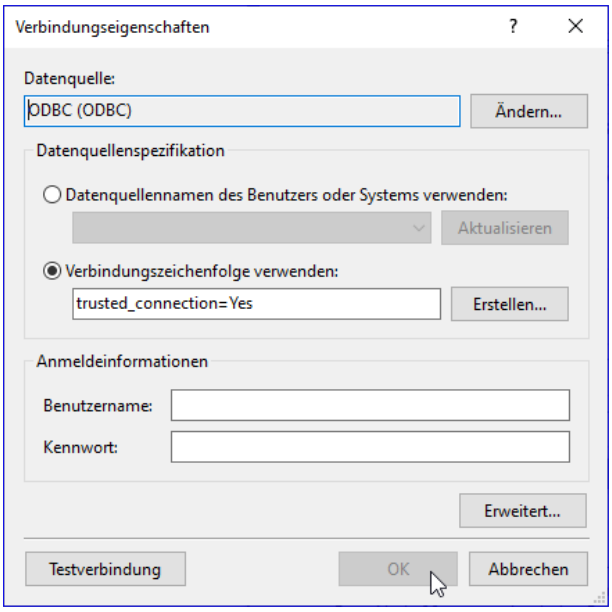


Bild 4: Verbindungseigenschaften für eine ODBC-Datenquelle

Datenquellen vor. Mit einem Rechtsklick auf diesen Ordner zeigen Sie den Kontextmenübefehl **Neue Datenquelle hinzufügen** an (siehe Bild 1).

Freigegebene Datenquellen in Visual Studio erstellen

Anschließend erscheint der Dialog **Eigenschaften der freigegebenen Datenquelle** (siehe Bild 2). Hier geben Sie den Namen der Datenquelle ein und wählen den Typ aus – in diesem Fall **Microsoft SQL Server**. Anschließend klicken Sie auf die Schaltfläche **Erstellen...**, um die Details festzulegen.

Dies liefert den Dialog **Verbindungseigenschaften**, den Sie bereits von anderen Anwendungen kennen dürften. Dieser wird je nach Auswahl des Typs der Datenquelle mit anderen Eigenschaften angezeigt. Für eine SQL Server-Datenbank sieht der Dialog beispielsweise wie in Bild 3 aus.

Wenn Sie hingegen eine ODBC-Datenquelle nutzen wollen und dazu im Dialog **Eigenschaften der freigegebenen Datenquelle** den Eintrag **ODBC** im Feld **Typ** auswählen,

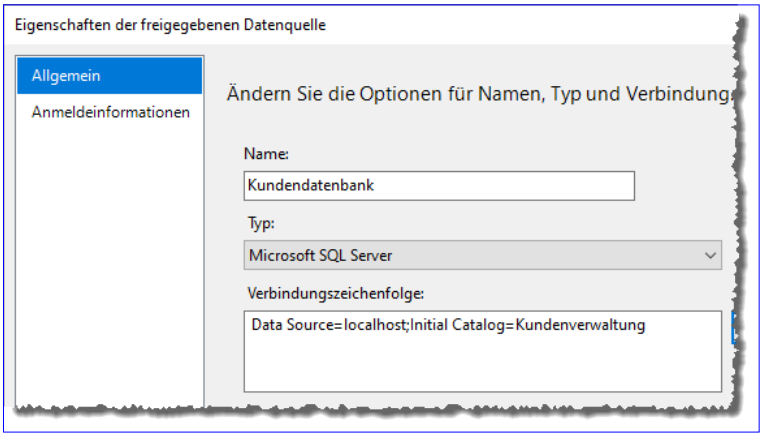


Bild 5: Verbindungszeichenfolge für eine SQL Server-Datenbank

erhalten Sie den Dialog aus Bild 4 für die Eingabe der Verbindungseigenschaften einer ODBC-Datenquelle.

Für manche Datenquellentypen gibt es auch gar keinen Dialog, sodass Sie die Verbindungszeichenfolge manuell eintragen müssen – beispielsweise beim Typ **XML**.

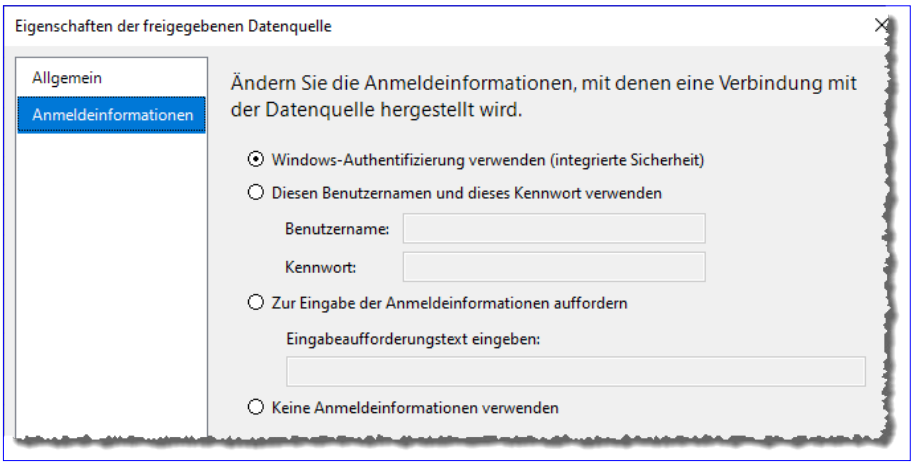


Bild 6: Einstellungen für die Anmeldeinformationen

SQL Server-Datenquelle

Wenn Sie mit dem Dialog **Verbindungseigenschaften** die Daten für eine SQL Server-Datenbank erfassen und das schnell erledigen wollen, haben Sie am besten den Namen des SQL Servers bereits parat. Das Ermitteln der verfügbaren Server mit der Schaltfläche Aktualisieren dauert immer eine Weile.

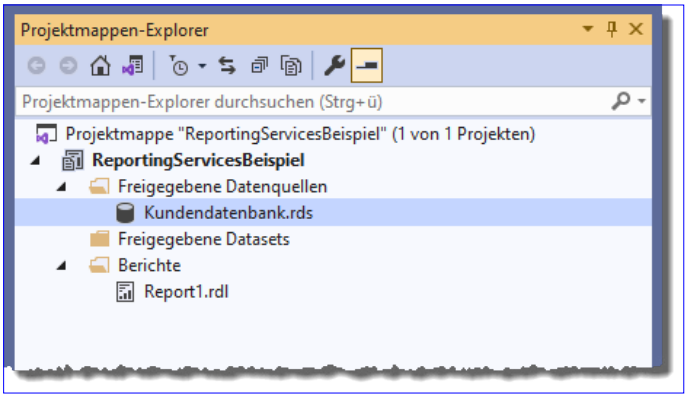


Bild 7: Neue Datenquelle für die Reporting Services

Wenn Sie jedoch den Namen direkt eintragen, indem Sie diesen entweder auswendig kennen (manchmal reicht ja auch **localhost**) oder aus dem SQL Server Management Studio kopieren, finden Sie schnell alle verfügbaren Datenbanken in der Liste unter **Mit Datenbank verbinden|Datenbanknamen auswählen oder eingeben** vor.

Wählen wir hier beispielsweise den Namen **Kundenverwaltung** aus und schließen den Dialog nach erfolgreichem Test mit der Schaltfläche **Testverbindung**, erhalten Sie die Verbindungszeichenfolge aus Bild 5.

Anmeldeinformationen festlegen

Anschließend stellen Sie im gleichnamigen Bereich noch die **Anmeldeinformationen** ein (siehe Bild 6). Hier behalten wir die Option **Windows-Authentifizierung verwenden (integrierte Sicherheit)** bei. Damit können wir die aktuellen Windows-Benutzer auf die verschiedenen Benutzergruppen/Benutzer mit den entsprechenden Berechtigungen mappen. Sie können auch Benutzername/Kennwort verwenden oder zur Eingabe von Anmeldeinformationen auffordern.

Nach dem Anlegen der Reporting Services Datenquelle

Haben Sie das Anlegen abgeschlossen, finden Sie die neue Datenquelle als Eintrag im Ordner **Freigegebene Datenquellen** vor (siehe Bild 7). Reporting Services Datenquellen haben eine Bezeichnung mit der Endung **.rds**.

Reporting Services Datasets

Wenn Sie ein Reporting Services Projekt erstellt haben, benötigen Sie vor dem Anlegen von Berichten zwei Dinge: eine Datenquelle, auch Datasource genannt, und eine Art Abfrage, wie bei den Reporting Services Datasets genannt werden. Wie Sie Datenquellen auf verschiedene Arten erstellen, haben wir im Artikel »Reporting Services Datenquellen« erläutert. Im vorliegenden Artikel beschreiben wir, wie Sie mithilfe von Datasets die Daten für Ihren Bericht zusammenstellen und welche unterschiedlichen Möglichkeiten es dafür gibt. Dabei schauen wir uns wieder Visual Studio, den SQL Server Report Builder und das Web-Portal der Reporting Services an.

Freigegebene Datasets oder eingebettete Datasets?

Genau wie bei den Datenquellen beziehungsweise Datasources gibt es auch bei den Datasets zwei Varianten – nämlich freigegebene Datasets und eingebettete Datasets. Die freigegebenen Datasets können Sie in einem Projekt oder im Web-Portal erstellen. Eine wichtige Voraussetzung für ein freigegebenes Dataset ist das Vorhandensein einer freigegebenen Datenquelle. Für ein eingebettetes Dataset können Sie eine freigegebene Datenquelle verwenden, aber auch eine eingebettete Datenquelle.

Welchen Typ von Dataset sollten Sie nun verwenden? Es verhält sich in einem Aspekt ähnlich wie bei Microsoft Access: Dort konnten Sie Abfragen als eigene Objekte speichern und diese dann Formularen über die Eigenschaft **Datensatzquelle** oder Listensteuerelementen über die Eigenschaft **Datensatzherkunft** zuweisen. Oder Sie haben die als Datenquelle zu verwendende Abfrage als SQL-Abfrage direkt für die jeweilige Eigenschaft hinterlegt. Beides hat Vor- und Nachteile: Wenn Sie die Abfrage als eigenes Objekt speichern, können Sie diese als Datenquelle für mehr als ein Formular oder einen Bericht und gleichzeitig noch als Datenquelle für Listensteuerelemente nutzen.

Der Vorteil und der Nachteil ist gleichzeitig, dass sich Änderungen an der Abfrage auf alle referenzierenden Elemente auswirkt. Andersherum sammeln Sie eine Menge Datenquellen in Formularen, Berichten und Steuerelementen an, die nur schwer zu warten sind.

Eine Alternative ist es, alle Abfragen als eigene Objekte zu speichern und diese den Objekten beziehungsweise Steuerelementen zuzuweisen. Freigegebene Datasets haben den Vorteil, dass diese gecached werden können.

Und es gibt noch einen gravierenden Unterschied von freigegebenen Dataset gegenüber den erwähnten Abfragen von Access: Sie können diese nämlich auch von anderen Projekten aus verwenden!

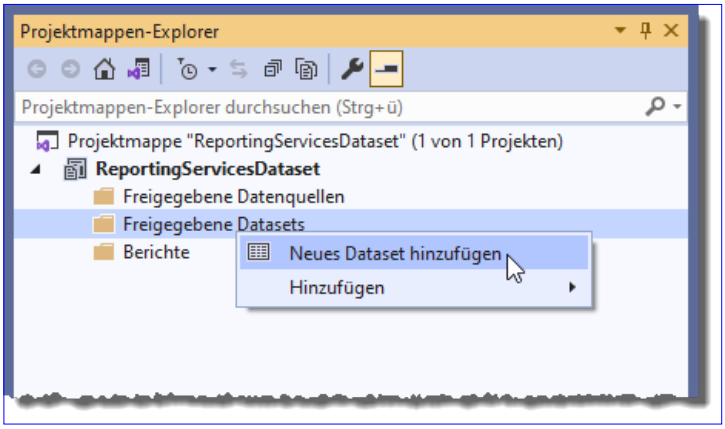


Bild 1: Hinzufügen eines neuen Datasets per Kontextmenü

Freigegebenes Reporting Services Dataset mit Visual Studio

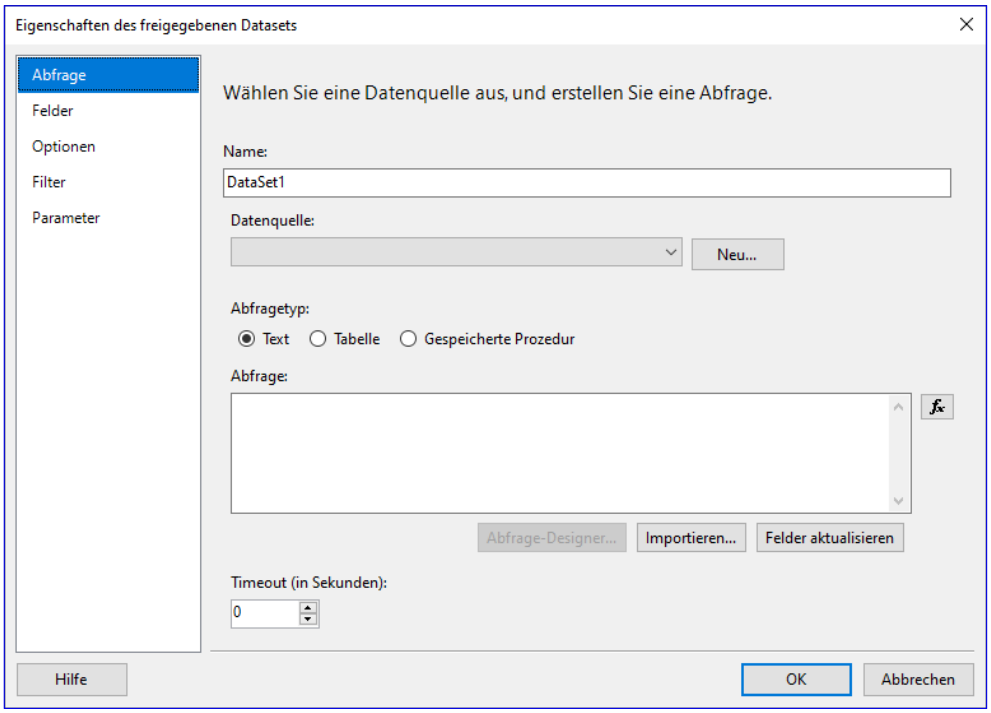


Bild 2: Dialog zum Anlegen eines neuen Datasets

Wenn Sie in Visual Studio ein neues Projekt des Typs **Berichtsserverprojekt-Assistent** oder **Berichtsserverprojekt** erstellt haben, finden Sie im Projektmappen-Explorer neben **Freigegebene Datenquellen** einen weiteren Ordner namens **Freigegebene Datasets** vor. Wollen Sie für diesen ein neues freigegebenes Dataset hinzufügen, betätigen Sie den Kontextmenü-Befehl **Neues Dataset hinzufügen** dieses Eintrags (siehe Bild 1). Hierbei ist es wichtig zu wissen, dass Sie auf jeden Fall zuvor mindestens eine freigegebene Datenquelle benötigen. Diese können Sie, wie im Artikel **Reporting Services Datenquellen (www.datenbankentwickler.net/259)** beschrieben erstellen – oder Sie klicken einfach auf **Neues Dataset hinzufügen** und legen die Datenquelle über die dortige Funktion an.

Der Kontextmenü-Eintrag öffnet den Dialog **Eigenschaften des freigegebenen Datasets**. Hier legen Sie den Namen des Datasets fest. Außerdem wählen Sie die Datenquelle aus den freigegebenen Datenquellen aus. Sollte noch keine freigegebene Datenquelle vorhanden sein oder wollen Sie eine neue Datenquelle erstellen, können Sie das mit einem Klick auf die Schaltfläche **Neu...** erledigen. Das liefert den Dialog **Eigenschaften der freigegebenen Datenquelle**, den wir im oben genannten Artikel im Detail beschrieben haben (siehe Bild 2).

An dieser Stelle gehen wir davon aus, dass Sie gemäß der Anleitung in dem oben erwähnten Artikel bereits eine Datenquelle erstellt haben.

Nun schauen wir uns den Abfragetyp an. Hier gibt es drei Möglichkeiten:

- **Text**: Eingabe einer Abfrage im SQL-Format. Dazu können Sie verschiedene Tools nutzen, die wir im Anschluss vorstellen werden.

- **Tabelle**: Erlaubt die Auswahl einer Tabelle. Diese Option steht jedoch nur in ODBC- oder OLE DB-Datenquellen zur Verfügung.

- **Gespeicherte Prozedur**: Wenn Sie die Daten für den Bericht bereits in einer gespeicherten Prozedur vorbereitet haben, können Sie diese aus der Liste der gespeicherten Prozeduren der in der Datenquelle angegebenen Datenbank auswählen.

SQL-Abfrage als Quelle für ein Dataset definieren

Wenn Sie unter Abfragetyp die Option **Text** wählen, können Sie verschiedene Optionen zum Erstellen der Abfrage nutzen:

- **Abfrage-Designer...**: Dies öffnet den Abfrage-Designer, mit dem Sie ähnlich wie im Abfrageentwurf von Access die Datenquelle zusammenklicken können.

- **Importieren...**: Öffnet einen Dateiauswahl-Dialog, mit dem Sie eine zuvor gespeicherte Abfragedatei mit der Dateiendung **.sql** oder **.rdl** auswählen können.

- **Felder aktualisieren**: Aktualisiert die Liste der Felder im Bereich **Felder** des Dialogs.

- **fx**: Diese Schaltfläche öffnet einen Dialog, mit dem Sie den aktuellen Ausdruck bearbeiten und um weitere Informationen anreichern können.

SQL-Abfrage per Abfrage-Designer erstellen

Wählen Sie die Option **Abfrage-Designer...**, erscheint der Dialog aus Bild 3. Mit diesem fügen Sie zunächst die Tabellen hinzu, deren Daten im Bericht angezeigt werden sollen. Dann markieren Sie die Felder, die im Abfrageergebnis erscheinen sollen. Mit einem Klick auf die Ausrufezeichen-Schaltfläche können Sie das Abfrageergebnis vor-

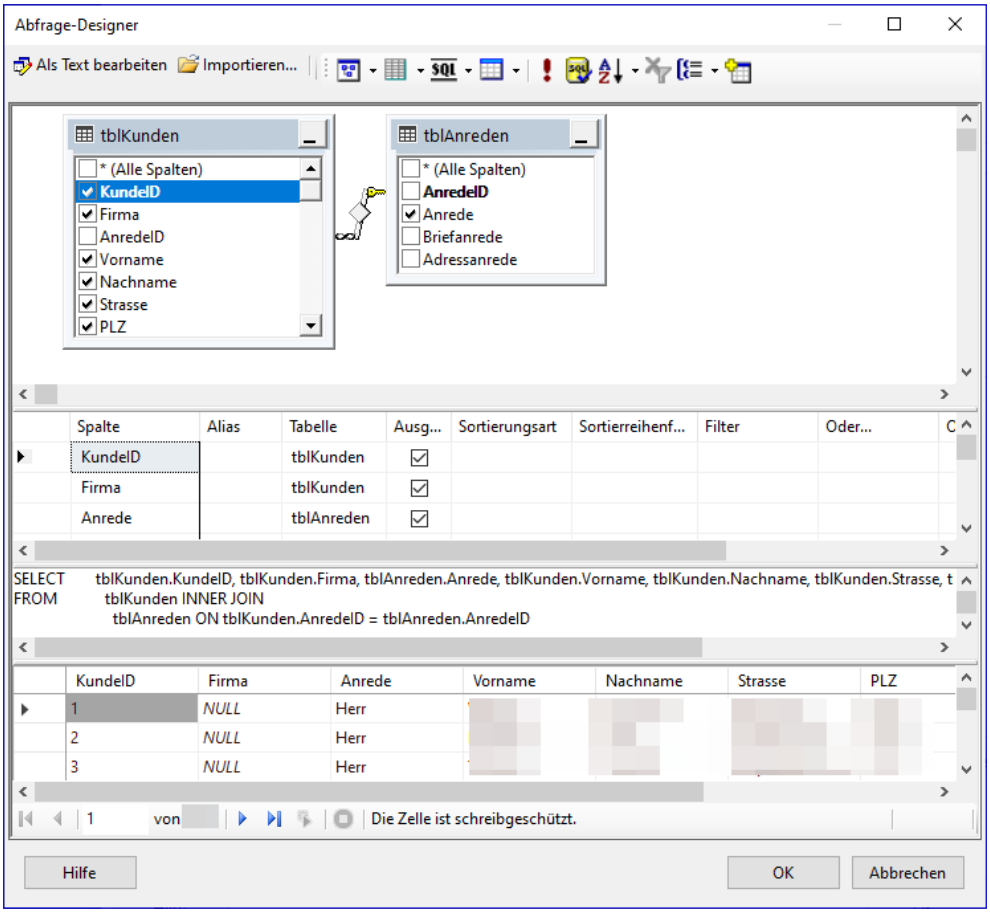


Bild 3: Entwurfsansicht eines Datasets

ab anzeigen lassen. In der Liste der anzuzeigenden Felder können Sie noch Sortierungen und Filterkriterien festlegen.

SQL-Abfrage bearbeiten

Die zuvor zusammengestellte SQL-Abfrage können Sie durch einen Klick auf die Schaltfläche mit der Beschriftung **fx** im Dialog **Ausdruck** anzeigen (siehe Bild 4). Dieser Dialog zeigt den SQL-Ausdruck in einem Textfeld an, wo Sie diesen manuell bearbeiten können.

Darunter finden Sie eine Liste mit Kategorien, nach deren Auswahl die Elemente der jeweiligen Kategorie rechts in der Liste mit der Beschriftung **Element** angezeigt werden. Diese können Sie allerdings an dieser Stelle nicht nutzen, da ein Hinzufügen nur per Doppelklick auf das jeweilige Element möglich ist, was aber dann den aktuellen Inhalt des Ausdrucks durch den angeklickten Eintrag ersetzt.

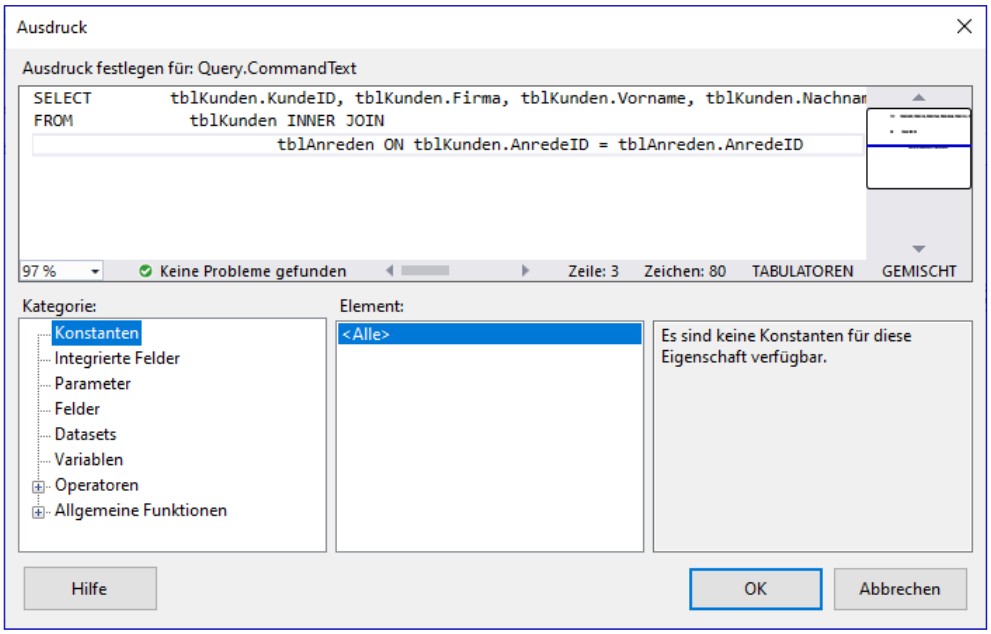


Bild 4: Bearbeiten des Abfrage-Ausdrucks

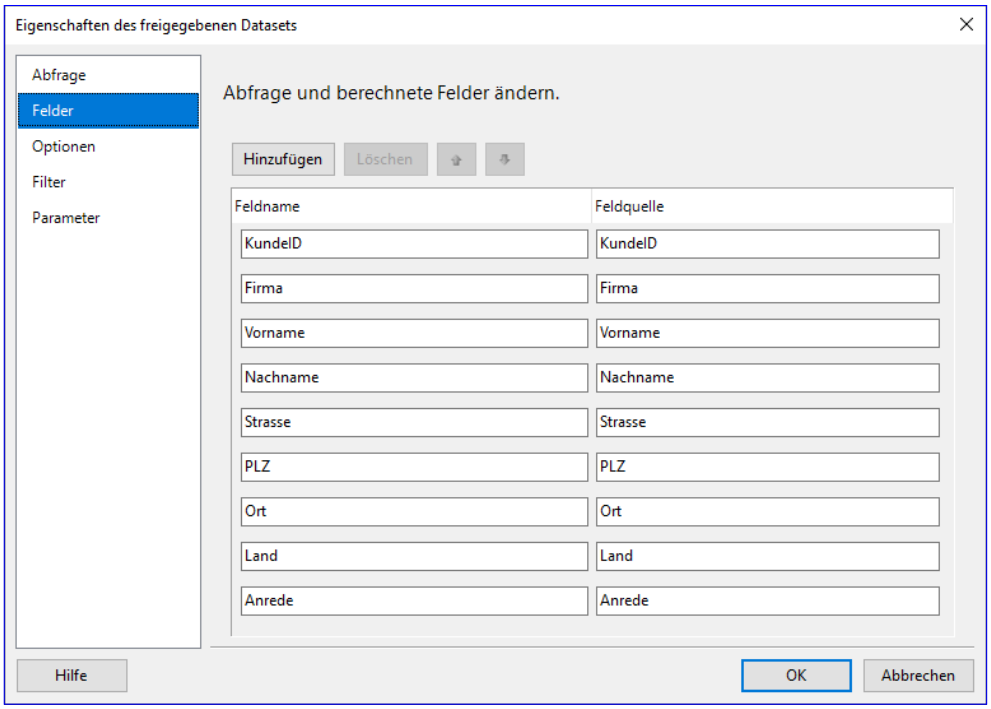


Bild 5: Bearbeiten der Felder der Abfrage

Der Bereich Felder in den Dataset-Eigenschaften

Wechseln Sie im Fenster **Eigenschaften des freigegebenen Datasets** zum Bereich **Felder**, zeigt der Dialog die Liste der Feldnamen und eine dazu passende Feldquelle an (siehe Bild 5). Hier finden Sie die über den Abfrage-

entwurf hinzugefügten Felder. Sie können aber noch weitere Felder hinzufügen und damit weitere Informationen für die Verarbeitung in den Berichen bereitstellen. Dazu stellt der Dialog verschiedene Schaltflächen zur Verfügung, zum Beispiel **Hinzufügen** und **Löschen** und zwei Schaltflächen zum Ändern der Reihenfolge der angezeigten Felder.

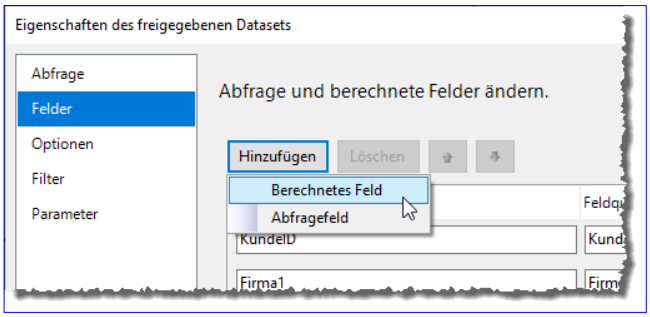


Bild 6: Hinzufügen von Feldern zum Dataset

Klicken Sie auf die Schaltfläche **Hinzufügen**, erscheinen zwei weitere Befehle (siehe Bild 6):

- **Berechnetes Feld**: Fügt eine neue Zeile hinzu, mit der Sie über die Schaltfläche fx den gewünschten berechneten Ausdruck zusammenstellen können.
- **Abfragefeld**: Fügt ein neues Feld hinzu, das Sie manuell ausfüllen müssen.

Berechnete Felder zu Dataset hinzufügen

Ein neues, berechnetes Feld sieht wie in Bild 7 aus.

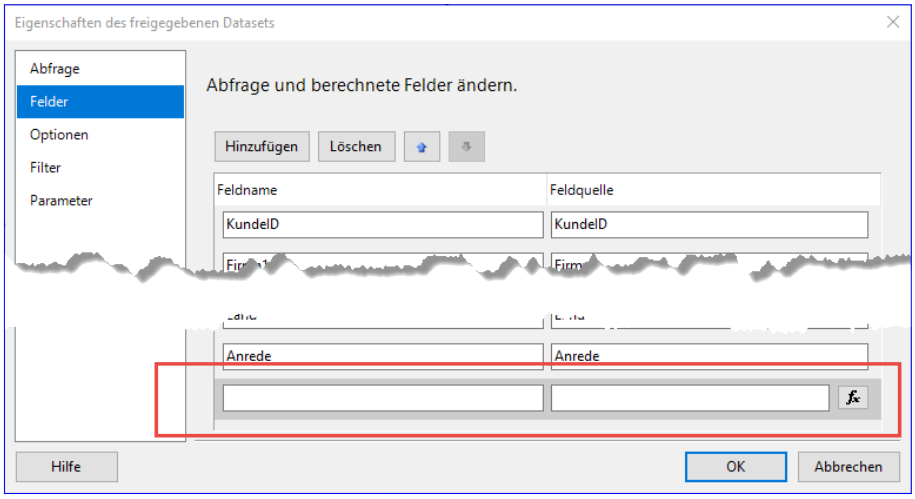


Bild 7: Einfügen eines berechneten Feldes

Klicken Sie auf die Schaltfläche **fx**, erscheint der Dialog **Ausdruck**. An dieser Stelle, also beim Anlegen oder Bearbeiten eines Datasets, sind die Möglichkeiten dieses Dialogs begrenzt, denn das Dataset ist noch keinem Bericht zugeordnet. Daher können wir, auch wenn wir bereits eine SQL-Abfrage als Datenquelle festgelegt haben, beispielsweise noch nicht auf die Felder dieser Abfrage zugreifen, um diese in Ausdrücken zu verwenden.

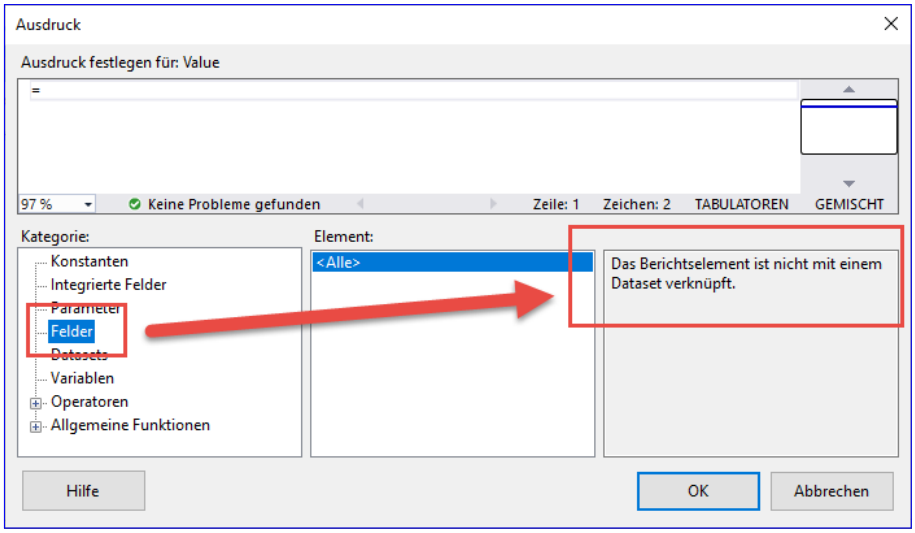


Bild 8: Für freigegebene Datasets fehlen wichtige Elemente.

Daher erscheint beispielsweise für die Kategorie Felder der Text **Das Berichtselement ist**

nicht mit einem Dataset verknüpft (siehe Bild 8).

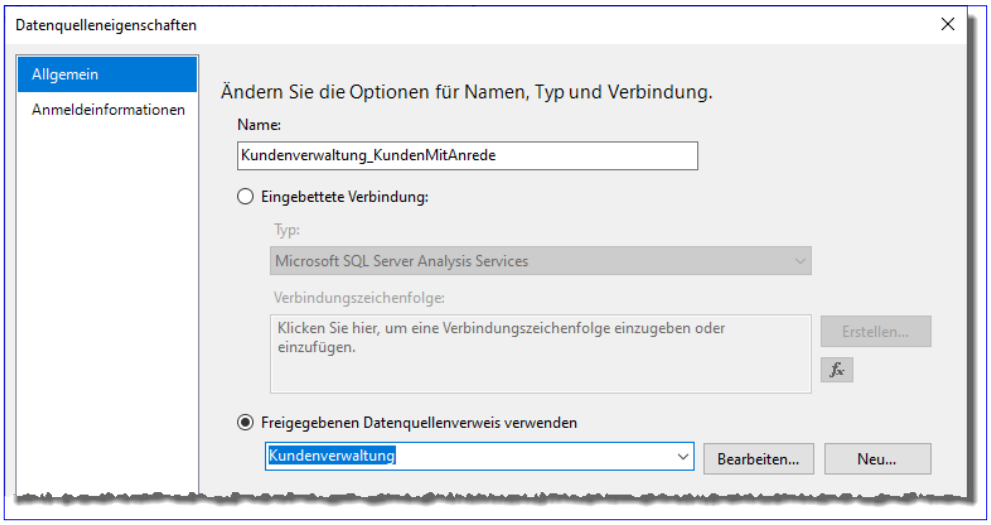


Bild 9: Hinzufügen der eingebetteten Datenquelle zu einem Bericht

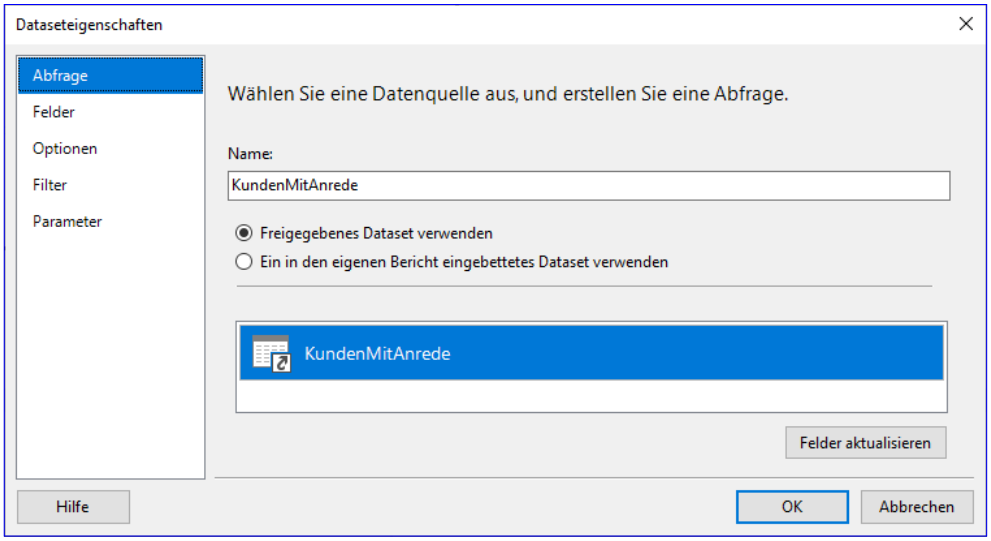


Bild 10: Auswählen des freigegebenen Datasets als Dataset des Berichts

Also machen wir einen Zwischenschritt und erstellen einen neuen Bericht. Dazu wählen wir den Kontextmenü-Eintrag **Hinzufügen|Neues Element...** des Projekt-Elements im Projektmappen-Explorer aus und legen im Dialog **Neues Element hinzufügen** einen neuen Bericht namens **KundenMitAnrede.rdl** an. Für diesen Bericht fügen wir dann im Bereich **Berichtsdaten** (einzublenden mit dem Menübefehl **Ansicht|Berichtsdaten**) eine neue Datenquelle hinzu. Das erledigen wir mit dem Kontextmenü-Befehl **Datenquelle hinzufügen...** des Ordners **Datenquellen**. Die neue Datenquelle soll **Kundenverwaltung_KundenMitAnrede** heißen und die freigegebene Datenquelle **Kundenverwaltung** nutzen (siehe Bild 9).

Nach dem Schließen des Dialogs öffnen wir über den Kontextmenü-Befehl **Dataset hinzufügen...** des Ordners **Datasets** den Dialog **Dataseteigenschaften**. Dieser zeigt direkt das einzige freigegebene Dataset der verwendeten freigegebenen Datenquelle an, die wir nun auswählen (siehe Bild 10).

Wo der Dialog bereits geöffnet ist, wechseln wir direkt zum Bereich **Felder**, der uns gerade noch nicht alle Funktionen offenbart hat. Klicken wir dort auf **Hinzufügen|Berechnetes Feld** und dann für das neue Feld auf die Schaltfläche **fx**, finden wir unter der Kategorie **Felder** immer noch nicht die Felder der Abfrage vor. Dazu müssen wir den Dialog erst schließen und so das Dataset dem aktuellen Bericht zuweisen. Erst nach einem erneuten Öffnen des Datasets, dem Anklicken von **Hinzufügen|Berechnetes Feld** im Bereich **Felder** und einem Klick auf die Schaltfläche **fx** für das neue

Element, finden wir unter der Kategorie **Felder** die Felder der Abfrage vor (siehe Bild 11). Damit können wir nun Ausdrücke auf Basis der Felder zusammenstellen. So können wir beispielsweise PLZ und Ort zu einem einzigen Feld zusammenführen:

```
=Fields!PLZ.Value & " "
& Fields!Ort.Value
```

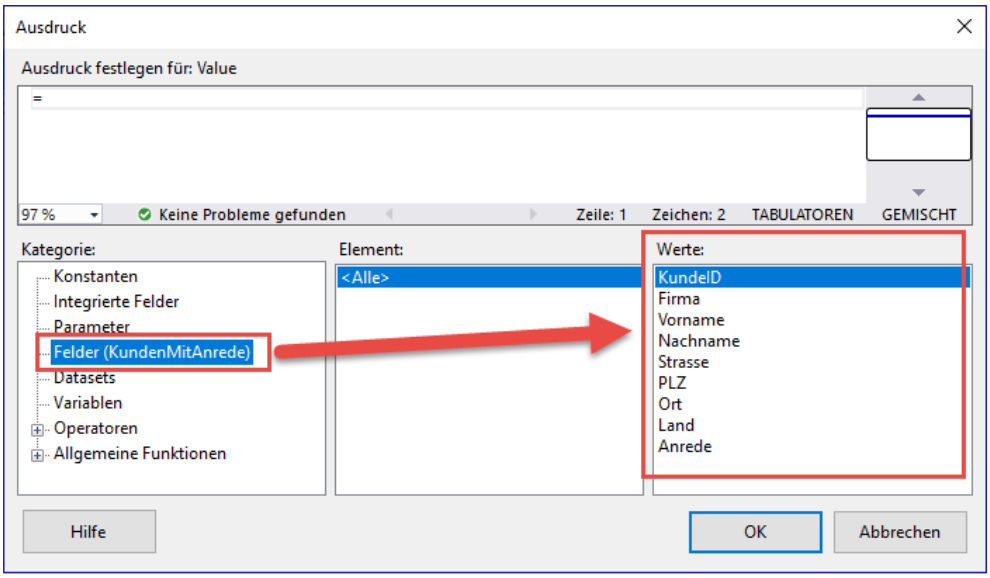


Bild 11: Zugriff auf die Felder eines Datasets für das Definieren eines Ausdrucks

Im Dialog **Ausdruck positionieren** Sie dazu die Einfügemarke jeweils an der entsprechenden Stelle unter **Ausdruck festlegen für: Value** und klicken dann doppelt auf die Elemente der Liste **Werte:**, um diese zum Ausdruck hinzuzufügen (siehe Bild 12). Im Ausdruck können Sie auch Visual Basic-Befehle verwenden. So können Sie beispielsweise Teile von Datumsfeldern mit Funktionen wie **Day**, **Month** oder **Year** extrahieren.

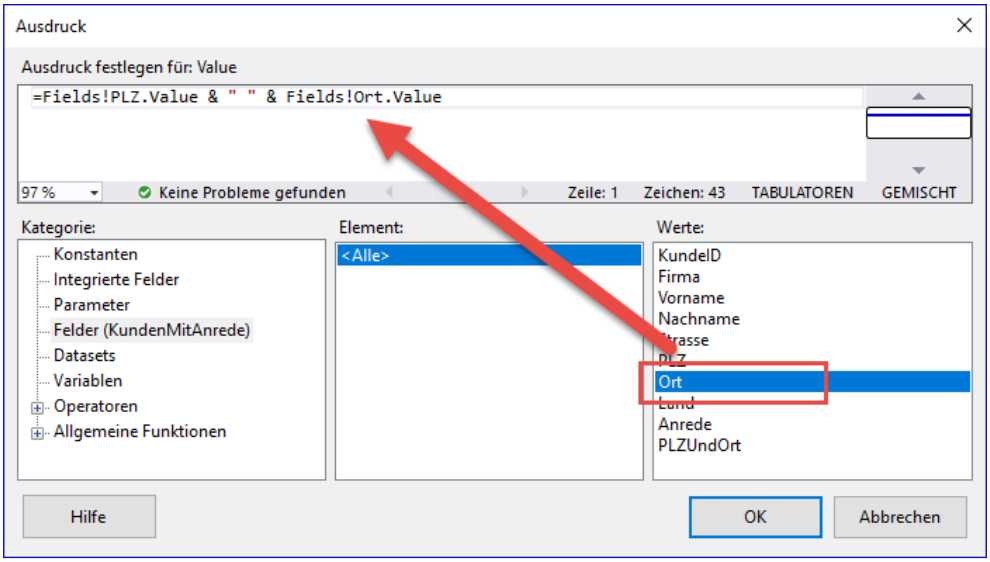


Bild 12: Zusammenstellen eines berechneten Ausdrucks

Benutzerdefinierte Visual Basic-Funktionen in Ausdrücken

Sie können sogar eigene Visual Basic-Funktionen definieren und diese zur Berechnung von Ausdrücken heranziehen. Diese Möglichkeit werden wir in einem Artikel namens **Reporting Services Berichte mit VB-Funktionen** (**www.datenbankentwickler.net/264**) vorstellen.

Filter verwenden

Im Bereich **Filter** des Dialogs **Dataseteigenschaften** legen Sie Filterkriterien für das Dataset fest. Hier finden Sie zunächst nur die aktivierte Schaltfläche **Hinzufügen** vor. Diese zeigt einen neuen, leeren Block für ein Filter-

Reporting Services Datasets mit Parametern

Ein Dataset, das als Datenquelle eines Berichts der Reporting Services dient, ist selten statisch aufgebaut – also so, dass es immer alle Daten einer Tabelle oder Abfrage liefert oder einen bestimmten Teil. Stattdessen sollen Berichte oft Daten liefern, die sich auf einen bestimmten Zeitraum beziehen, auf einen bestimmten Kunden oder auch auf bestimmte Artikel oder Kategorien. In den bisherigen Artikeln haben wir erläutert, wie Sie Berichte auf Basis statischer Datenquellen erstellen. Der vorliegende Artikel fügt die Verwendung von Parametern mit hinzu, damit der Benutzer genau festlegen kann, welche Daten der von ihm gewählte Bericht liefern soll.

Vorbereitung für diesen Artikel

Da wir uns explizit um den Einsatz von Parametern kümmern wollen, gehen wir von einem Visual Studio-Projekt des Typs **Berichtsserverprojekt** namens **DatasetMitParametern** aus.

Diesem haben wir eine freigegebene Datenquelle namens **AdventureWorks** auf Basis der SQL Server-Datenbank **AdventureWorksLT2019** hinzugefügt. Wo Sie diese Beispieldatenbank finden und wie Sie diese installieren, erfahren Sie im Artikel **AdventureWorks: Schnelle Beispieldatenbank (www.datenbankentwickler.net/254)**.

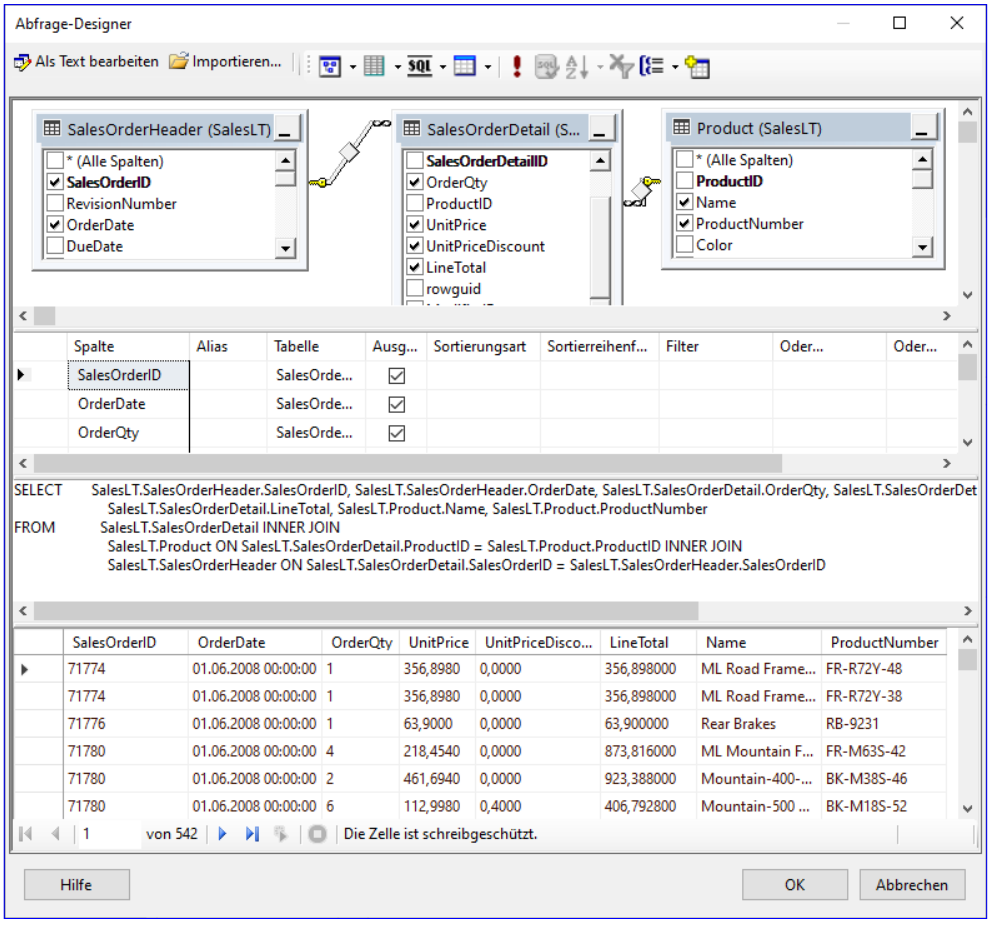


Bild 1: Abfrage für das Dataset **ProductsSalesOrders**

Dataset anlegen

Das Dataset für diesen Bericht legen wir als eingebettetes Dataset an. Damit erhalten wir zum Beispiel den Vorteil, dass die in diesem Dataset definierten Parameter direkt vom Bericht als solche erkannt werden.

Bericht anlegen

Nun legen wir einen neuen Bericht namens **SalesPerProduct.rdl** an. Diesem weisen wir die zuvor angelegte freigegebene Datenquelle **AdventureWorks.rds** als Datenquelle zu und geben auch der neuen eingebetteten Datenquelle den Namen **AdventureWorks**.

Dann legen wir für den Bericht ein neues eingebettetes Dataset namens **ProductsSalesOrders** an (siehe Bild 1). Das Dataset verwendet einige Felder der drei Tabellen **Product**, **SalesOrderDetail** und **SalesOrderHeader** als Datenquelle. Damit haben wir ausreichend Daten zum Demonstrieren der Parameter-Funktionen der Reporting Services.

Der Bereich **Berichtsdaten** sieht anschließend wie in Bild 2 aus.

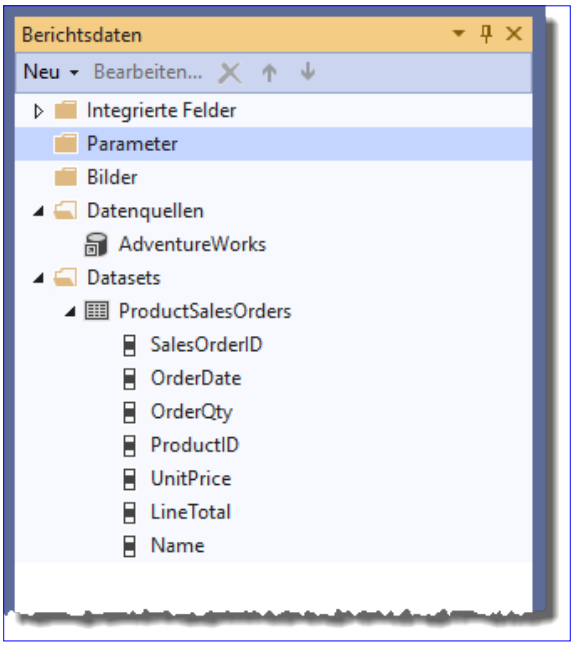


Bild 2: Daten für den Bericht

Nun blenden wir die Toolbox ein und fügen dem Bericht ein **Tabelle**-Element hinzu. Dann wechseln wir wieder zum Berichtsdaten-Bereich und ziehen einige Felder in die Tabelle, sodass diese wie in Bild 3 aussieht.

Ein Wechsel in die Vorschauansicht liefert eine Liste der einzelnen Bestellpositionen (siehe Bild 4). Damit haben wir alles zusammen, um die angezeigten Datensätze nach verschiedenen Kriterien mit noch zu ermittelnden Parameterwerten zu filtern.

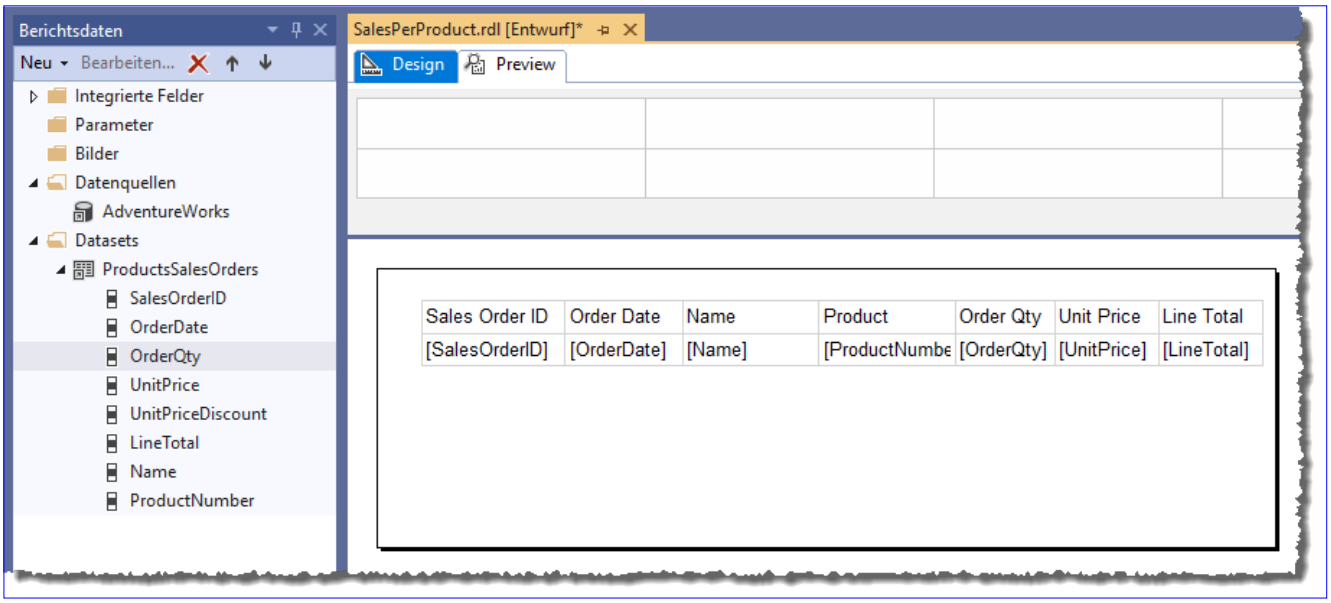
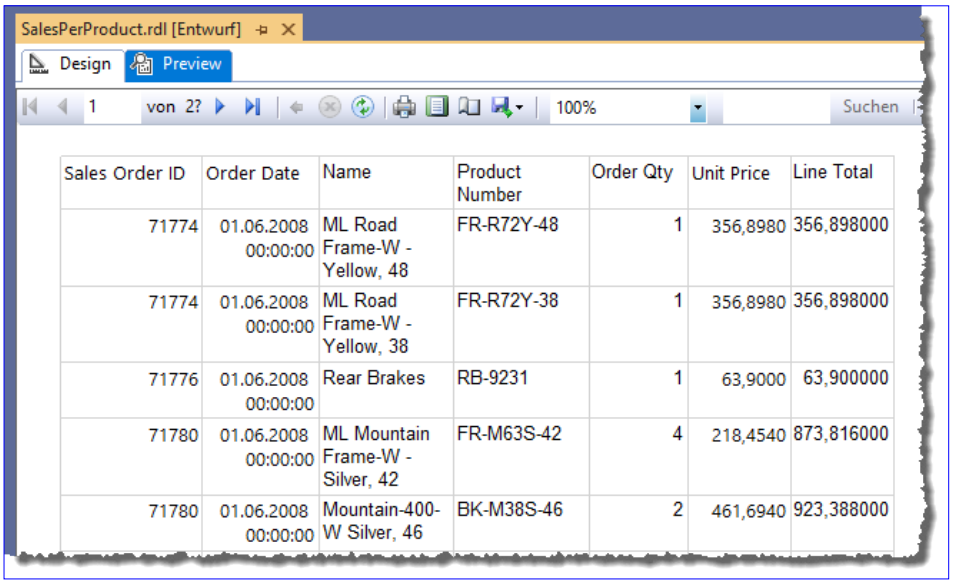


Bild 3: Berichtsentwurf

Parameter zum Dataset hinzufügen

Der erste Schritt ist das Hinzufügen des Parameters zum Dataset. Dazu öffnen wir das Dataset per Doppelklick auf seinen Eintrag im Bereich Berichtsdaten.



Sales Order ID	Order Date	Name	Product Number	Order Qty	Unit Price	Line Total
71774	01.06.2008 00:00:00	ML Road Frame-W - Yellow, 48	FR-R72Y-48	1	356,8980	356,898000
71774	01.06.2008 00:00:00	ML Road Frame-W - Yellow, 38	FR-R72Y-38	1	356,8980	356,898000
71776	01.06.2008 00:00:00	Rear Brakes	RB-9231	1	63,9000	63,900000
71780	01.06.2008 00:00:00	ML Mountain Frame-W - Silver, 42	FR-M63S-42	4	218,4540	873,816000
71780	01.06.2008 00:00:00	Mountain-400-W Silver, 46	BK-M38S-46	2	461,6940	923,388000

Bild 4: Ausgabe der einzelnen Bestellpositionen

Im Bereich **Abfrage** des Dialogs **Dateseteigenschaften** finden wir im Feld **Abfrage** die SQL-Abfrage. Diese erweitern wir wie in Bild 5 um die folgende **WHERE**-Klausel:

```
WHERE SalesOrderDetail.ProductID = @ProductID
```

Um die Funktionsweise dieses Parameters auszuprobieren, klicken Sie auf die Schaltfläche **Abfrage-Designer...** und öffnen so den Dialog **Abfrage-Designer**. Hier finden Sie die neu hinzugefügte **WHERE**-Klausel ebenfalls vor. Um diese auszuprobieren, betätigen Sie die Schaltfläche mit dem roten Ausrufezeichen. Dadurch erscheint der Dialog **Abfrageparameter** (siehe Bild 6). Hier geben Sie den Wert ein, nach dem das Feld **ProductID** gefiltert werden soll, zum Beispiel **991**.

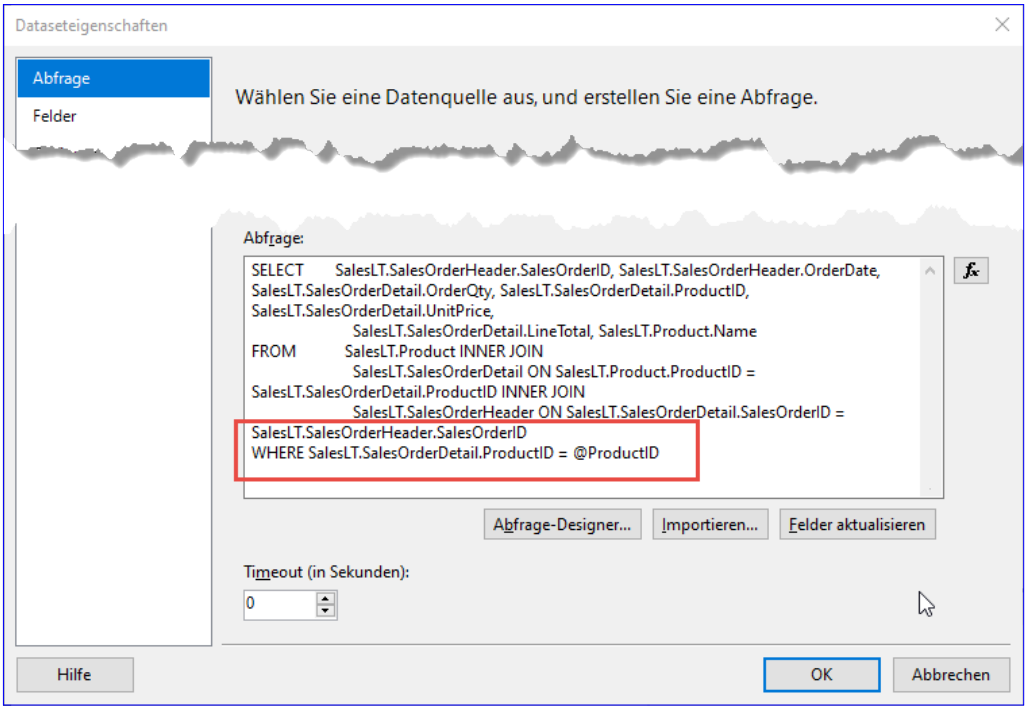


Bild 5: Integrieren des Parameters im Abfrageausdruck

Der Abfrage-Designer zeigt dann im unteren Bereich alle passenden Datensätze an. Nun wollen wir dafür sorgen, dass der Para-

meter auch beim Anzeigen des Berichts abgefragt und berücksichtigt wird.

Vorher werfen wir jedoch noch einen Blick auf den Bereich **Berichtsdaten**. Hier finden wir nämlich nun einen neuen Eintrag namens [@] **ProductID** vor (siehe Bild 7).

Eigenschaften des neuen Parameters festlegen

Den neuen Parameter wollen wir direkt anpassen. Die Eigenschaften des neuen Parameters zeigen Sie an, indem Sie doppelt auf den Eintrag im Ordner **Parameter** klicken. Dies öffnet den Dialog **Berichtsparametereigenschaften** (siehe Bild 8).

Der erste Bereich namens **Allgemein** erwartet die folgenden Informationen:

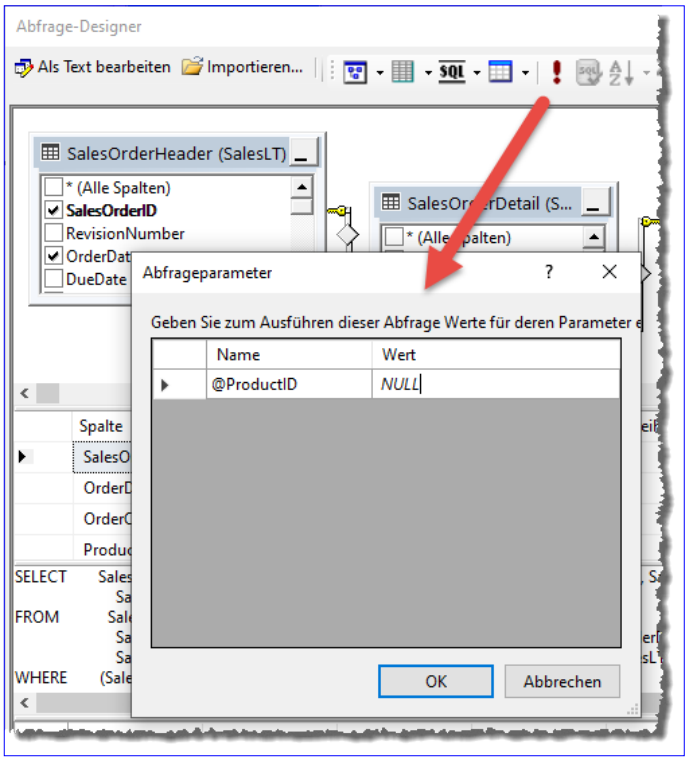


Bild 6: Eingabe des Parameterwertes

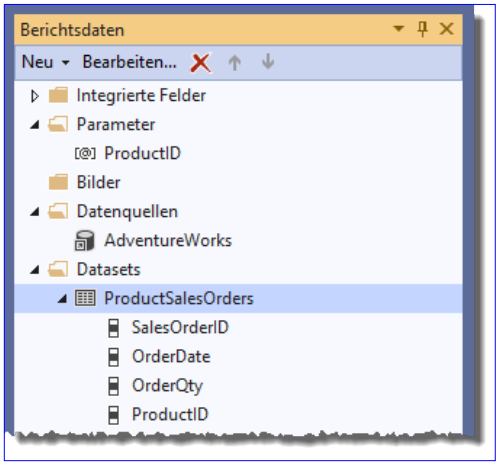


Bild 7: Neuer Eintrag im Ordner **Parameter**

- **Name**: Bezeichnung des Parameters, unter dem Sie diesen später referenzieren können. Dieser wurde bereits mit **ProductID** gefüllt.
- **Eingabeaufforderung**: Text, der erscheint, um den Wert des Parameters vom Benutzer abzufragen. Den vorausgefüllten Text ändern wir in **Filtern nach Produkt-ID:**.
- **Datentyp**: Datentyp des Parameters. Mögliche Werte sind **Text**, **Boolean**, **Datum/Uhrzeit**, **Ganze Zahl** oder **Gleitkommawert**. Wenn Sie **Datum/Uhrzeit** auswählen, bekommt der Benutzer ein entsprechendes Steuerelement zur Auswahl des Datums angezeigt. In diesem Fall benötigen wir den Wert **Ganze Zahl**.
- **Leeren Wert zulassen**: Gibt an, ob leere Zeichenketten angegeben werden können.
- **NULL-Wert zulassen**: Gibt an, ob NULL-Werte zulässig sind. Dies ist nötig, wenn Sie wie weiter unten beschrieben alle Datensätze des Dataset anzeigen wollen.
- **Mehrere Werte zulassen**: Gibt an, ob die Eingabe mehrerer Werte möglich sein soll. Dafür darf die vorherige Eigenschaft keine NULL-Werte zulassen. Später zeigen wir ein Beispiel für diese Einstellung.

Anzeige eines Textfeldes zur Parametereingabe

Neben dem neuen Eintrag im Ordner Parameter sehen wir außerdem nun, wozu die weißen Felder oben im Berichtsentwurf nützlich sind. Hier zeigen die Reporting Services nämlich die Parametertexte und die Steuerelemente zur Eingabe der Parameter an – in unserem Fall ein Feld mit der Beschriftung **Filtern nach Produkt-ID:**.

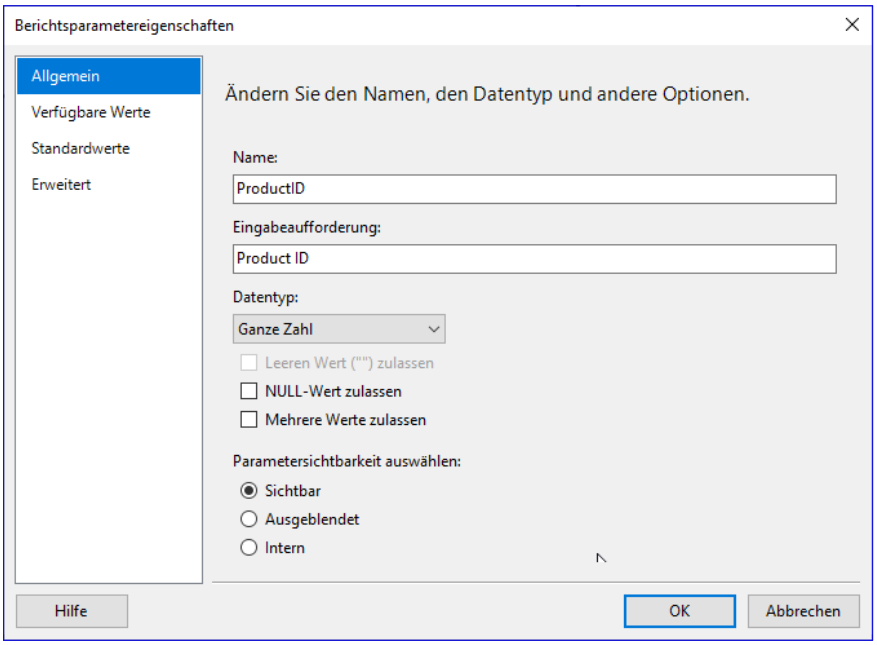


Bild 8: Eigenschaften des neuen Parameters

Wechseln wir nun in die Vorschau-Ansicht, bleibt dieses Eingabefeld erhalten und wir können nach der gewünschten **Produkt-ID** filtern (siehe Bild 9).

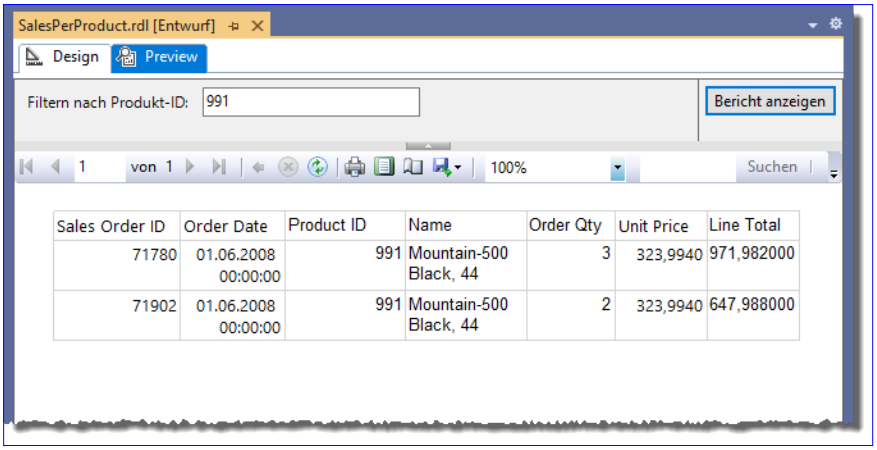


Bild 9: Eingeben des Parameterwertes und Filtern nach der Produkt-ID

Das sieht doch schon sehr benutzerfreundlich aus und viel schöner als beispielsweise die Parameter-Dialoge von Access, in denen man jeden Parameter einzeln eingeben muss. Hier können Sie den Parameterwert ändern und mit der Schaltfläche **Bericht anzeigen** oder einfach durch Betätigen der Eingabetaste die Anzeige aktualisieren.

Alle Datensätze anzeigen, wenn der Parameter NULL ist

Vielleicht möchten Sie, bevor Sie nach einem bestimmten Produkt filtern, erst einmal alle Datensätze des Datasets sehen. In diesem Fall müssen Sie zunächst die Abfrage des Datasets ändern. Die Änderung ist überschaubar – wir fügen lediglich ein weiteres Kriterium zum **WHERE**-Teil hinzu:

```
WHERE (SalesLT.SalesOrderDetail.ProductID = @ProductID OR @ProductID IS NULL)
```

Mit diesem prüfen wir, ob der Parameter **@ProductID** den Wert **NULL** aufweist. Sprich: Wenn der Benutzer keinen Parameter eingegeben hat, liefert die Abfrage alle Datensätze. Das können Sie auch gleich prüfen, indem Sie auf die Schaltfläche **Abfrage-Designer** klicken und im Abfrage-Designer die Abfrage mit einem Klick auf die

Schaltfläche mit dem roten Ausrufezeichen ausführen. Sie müssen nun noch sicherstellen, dass in den Eigenschaften des Parameters die Eigenschaft **NULL-Wert zulassen** aktiviert ist.

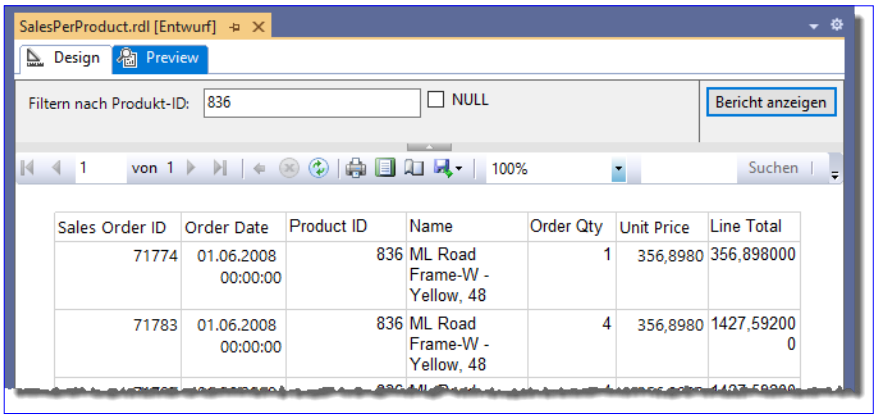


Bild 10: Möglichkeit, einen Parameter auf NULL einzustellen

Wenn Sie nun in die Vorschauansicht des Berichts wechseln, finden Sie neben dem Textfeld zur Eingabe der Produkt-ID noch ein Kontrollkästchen mit der Beschriftung **NULL** vor. Damit kann der Benutzer das Feld **Produkt-ID** leeren (siehe Bild 10).

Bericht mit Parametern veröffentlichen

Um den Bericht auf dem Reportserver und somit auch für andere Benutzer bereitzustellen, die nicht mit Visual Studio darauf zugreifen können, betätigen Sie in Visual Studio den Menübefehl **Erstellen|<Projektname> bereitstellen**. Wenn Sie danach mit **<Servername>/Reports** das Web-Portal anzeigen, finden Sie auf der ersten Seite einen Ordner für die Berichte des neuen Projekts vor (wenn der Report Server auf dem lokalen Rechner läuft, verwenden Sie **localhost** für **<Servername>**).

Klicken Sie den Ordner für das Projekt an und dann auf den Namen des Berichts, finden Sie den Bericht inklusive Steuerelementen für die Parametereingabe im Web-Portal vor (siehe Bild 11). Genau genommen sieht der Bereich zum Eingeben der Parameterwerte hier noch etwas professioneller aus, weil die freien Felder für weitere Parameter nicht angezeigt werden.

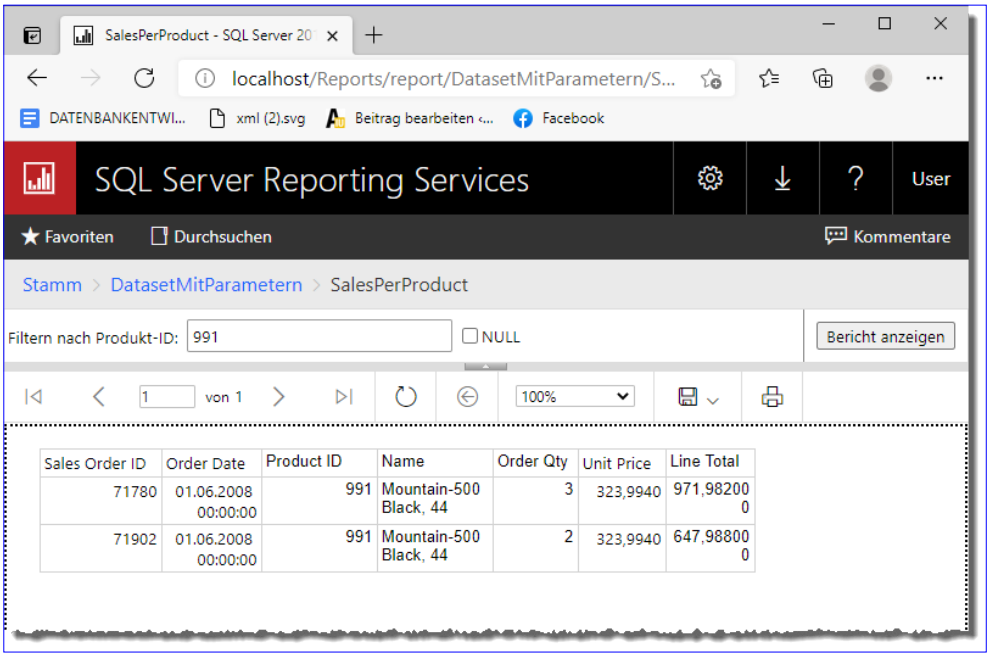


Bild 11: Bericht mit Parameter im Web-Portal

Bestelldatum als Parameter

Wir wollen noch einen weiteren Parameter hinzufügen – beziehungsweise gleich zwei.

Damit wollen wir das Datum der Bestellungen eingrenzen. Also wech-

Reporting Services: Tabellarische Berichte

Eine der einfachsten Arten, Daten in Berichten darzustellen, ist die tabellarische Darstellung. Die Reporting Services bieten beispielsweise gegenüber Microsoft Access eine erhebliche Vereinfachung: Sie können hier ein Steuerelement speziell zur Anzeige von Daten als Tabelle verwenden. Wie Sie dieses Steuerelement nutzen, und wie Sie auch mehrseitige Berichte mit allen notwendigen Informationen darstellen, zeigt der vorliegende Artikel.

Vorbereitung

An dieser Stelle gehen wir davon aus, dass Sie wie im Artikel **SQL Server Report Builder** (**www.datenbankentwickler.net/258**) den SQL Server Report Builder gestartet haben und dass sowohl eine Datenquelle als auch ein Dataset mit den abzubildenden Daten vorliegt (siehe Bild 1).

Die beiden standardmäßig in einem neuen Bericht enthaltenen Textfelder zum Hinzufügen eines Titels und der Ausführungszeit entfernen wir, indem wir diese markieren und die **Entf**-Taste betätigen. Dadurch erkennen wir, dass der Bericht eigentlich aus einem Berichts- und einem Fußzeile-Bereich besteht (siehe Bild 2).

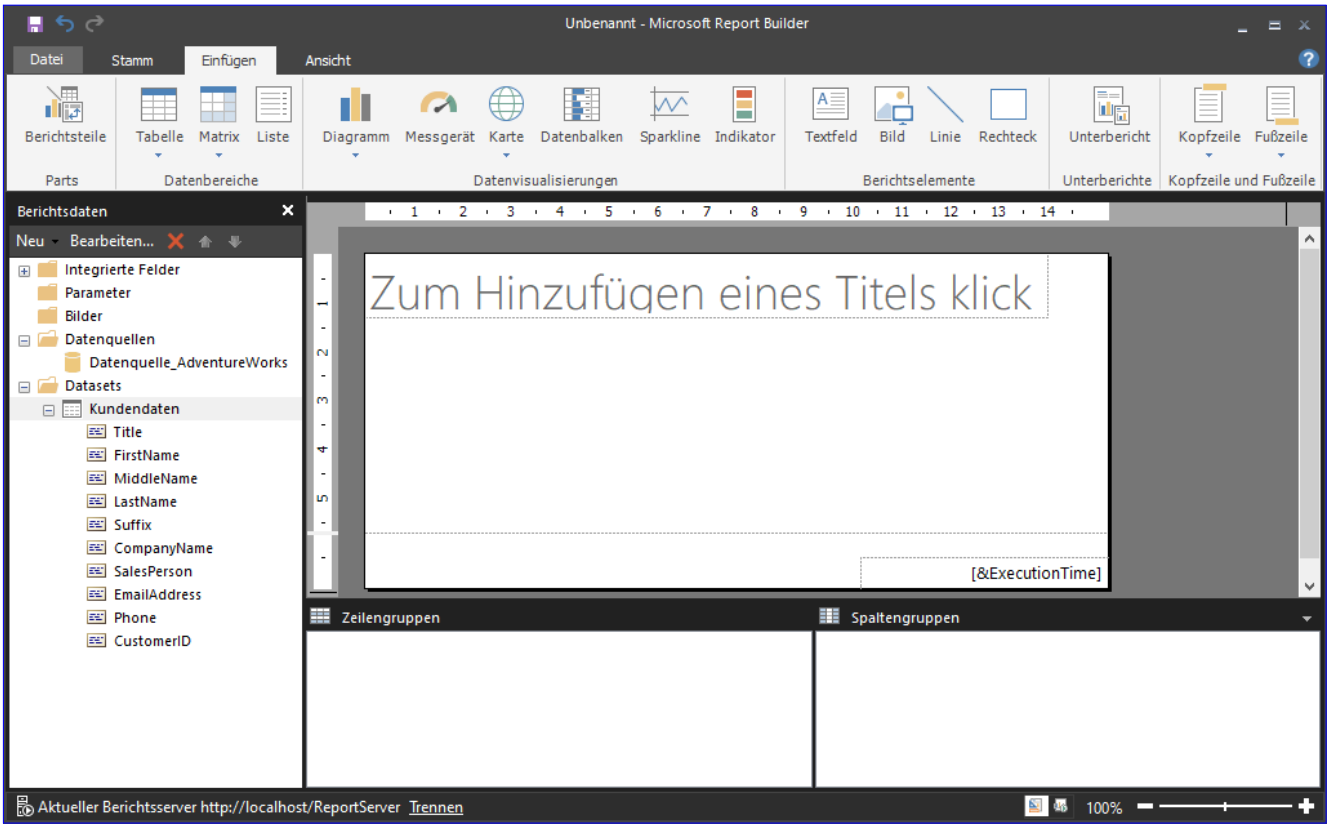


Bild 1: Ausgangssituation für die im vorliegenden Artikel beschriebenen Techniken

Eigenschaften einblenden

Wenn Sie Eigenschaften des aktuell markierten Elements im Berichtsentwurf jederzeit im Überblick haben wollen, wechseln Sie im Ribbon zum Bereich **Ansicht** und aktivieren dort die Option **Eigenschaften**. Hier können Sie außerdem noch den Bereich **Parameter** aktivieren (siehe Bild 3).

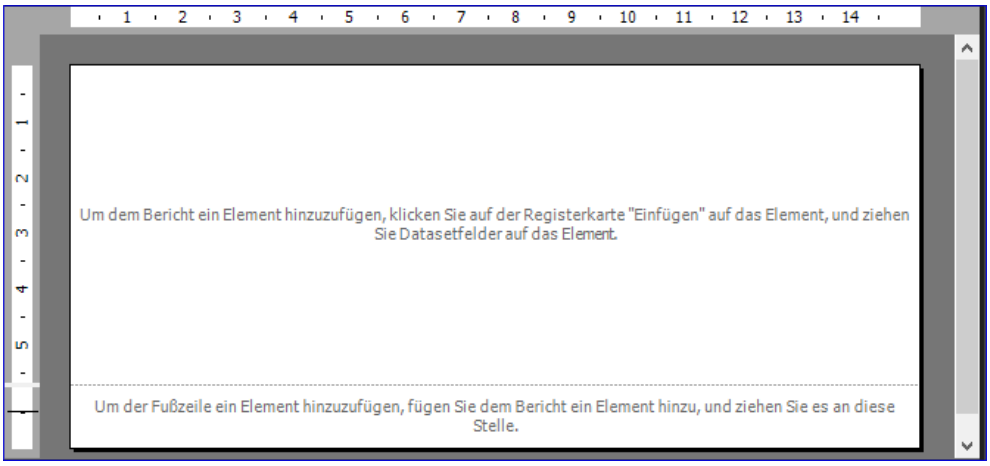


Bild 2: Entwurf des komplett geleerten Berichts

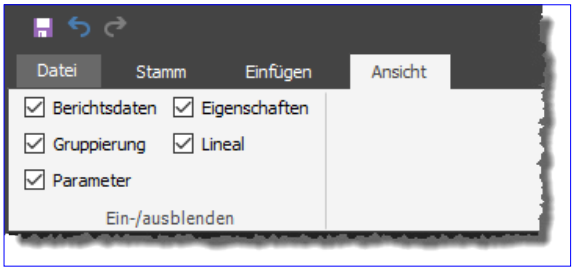


Bild 3: Bereiche ein- und ausblenden

Tabelle hinzufügen

Wir wollen erst einmal eine einfache Tabelle hinzufügen und uns ansehen, welche Möglichkeiten uns diese bietet. Steuerelemente fügen Sie auf folgende Arten hinzu:

- Indem Sie im Ribbon-Bereich **Einfügen** den Eintrag **Tabelle|Tabelle einfügen** anklicken und dann im Entwurf entweder auf den linken oberen Punkt der Tabelle klicken oder einen Rahmen mit der gewünschten Position und Größe aufziehen.

- Die zweite Möglichkeit ist ein rechter Mausklick in den Entwurf und die anschließende Auswahl des einzufügenden Steuerelements, hier also **Einsetzen|Tabelle** (siehe Bild 4).

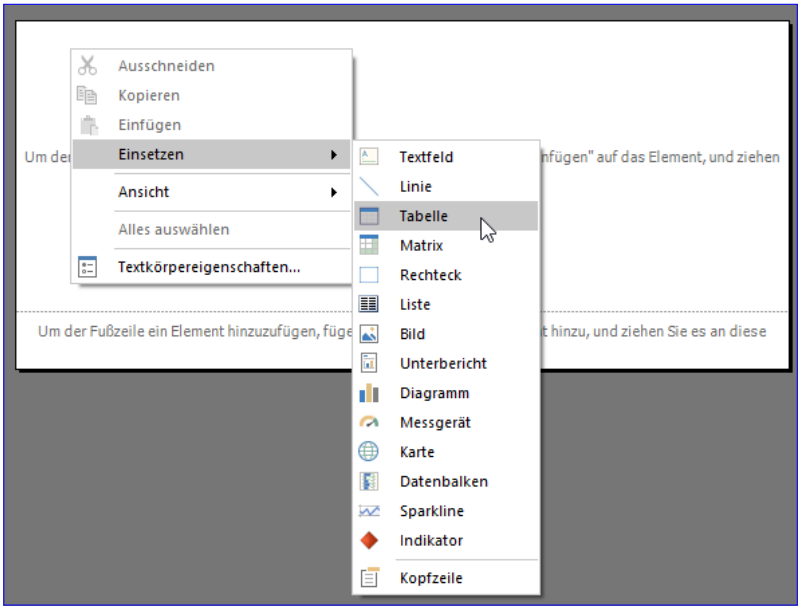


Bild 4: Hinzufügen einer Tabelle per Kontextmenü

Das Ergebnis ist eine neue Tabelle mit zwei Zeilen und drei Spalten, die nach dem Anklicken noch einige graue Kästen links und über den eigentlichen Zellen anzeigt (siehe Bild 5).

Felder hinzufügen per Drag and Drop

Sie können nun die gewünschten Felder aus dem Bereich Berichtsdaten direkt in

die gewünschten Spalten der Tabelle ziehen. Um ein Feld in eine Spalte zu ziehen, bewegen Sie dieses mit der Maus genau in die beiden übereinander angeordneten Felder, welche die Spalte ausmachen. Wenn das Plus-Zeichen im Mauszeiger erscheint, können Sie das Feld fallen lassen.

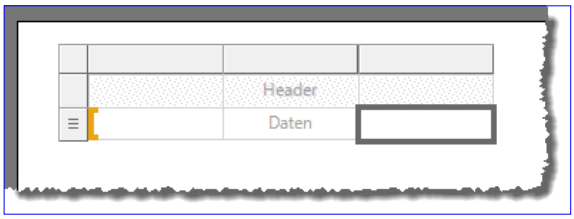


Bild 5: Eine neue, leere Tabelle

Die drei Spalten sind jedoch schnell voll. Was dann? Dann ziehen Sie weitere Felder einfach auf die schmalen Linien zwischen den Spalten beziehungsweise vor die erste oder hinter die letzte Spalte und lassen das Feld dann fallen. Auf diese Weise können Sie beliebig viele Felder hinzufügen, ohne zwischendurch mit anderen Methoden neue Spalten hinzufügen zu müssen.

Falls Sie dennoch einmal weitere Spalten benötigen, eine Spalte löschen wollen oder andere Operationen durchführen wollen, können Sie dazu das Kontextmenü einer Spalte nutzen (siehe Bild 6).

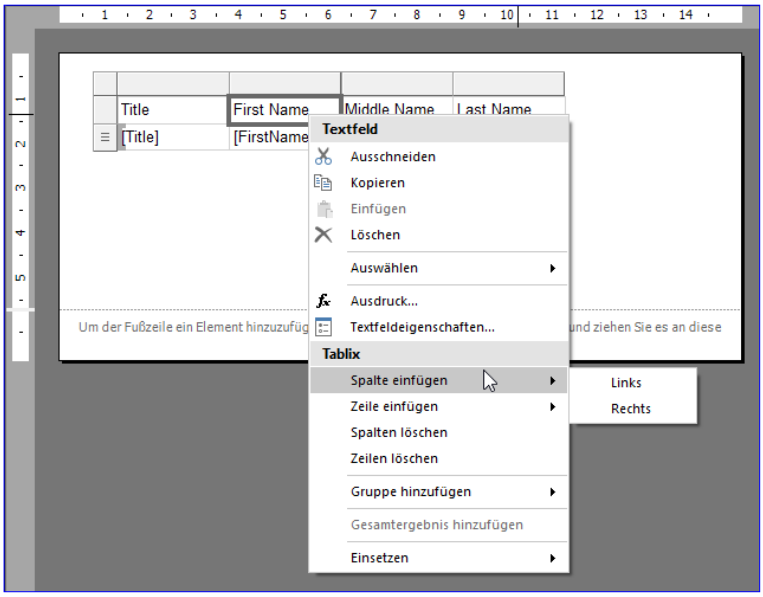


Bild 6: Einfügen neuer Spalten

Berichtsabmessungen

Auf diese Weise können Sie alle benötigten Felder zur Tabelle hinzufügen. Irgendwann reicht dann vielleicht der Platz in der Breite nicht mehr und Sie machen den weiß hinterlegten Bereich breiter. Hier gilt es dann zu beachten, welches Format für den Bericht ausgewählt ist und welche Seitenränder für diesen hinterlegt sind.

Die Informationen, die Sie nun benötigen, finden Sie im Bereich Eigenschaften, wenn Sie den Hintergrund des Berichts anklicken (siehe Bild 7). Hier finden Sie unter Seite zunächst die Eigenschaft **PageSize**. Dies entspricht in unserem Fall den Abmessungen einer DIN A4-Seite. Darunter finden Sie in der Eigenschaft **Margins** die vorgesehenen Seitenränder, in diesem Fall **2cm** an jeder Seite. Für die Breite des Berichts bedeutet das, dass Sie **21cm** minus zwei Mal **2cm** zur Verfügung haben, also **17cm**.

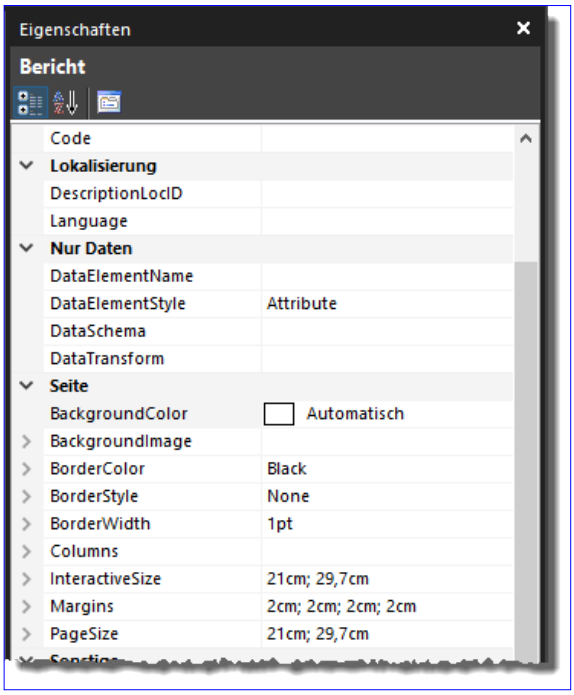


Bild 7: Abmessungen des Berichts und Seitenränder

Die Elemente sollten in der Breite also nicht mehr als **17cm** einnehmen, da sonst Teile der Tabelle auf einer weiteren Seite ausgegeben werden.

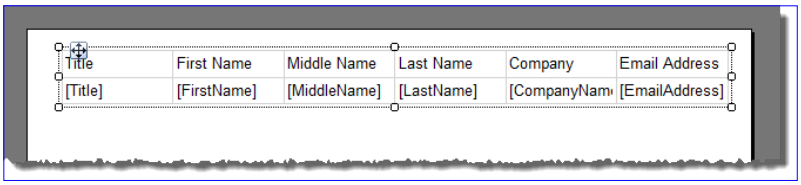


Bild 8: Verschieben einer Tabelle

Tabelle verschieben

Wenn Ihnen die Position der Tabelle noch nicht gefällt, können Sie diese verschieben. Das können Sie erledigen, sobald wie in Bild 8 oben links die Markierung zum Verschieben von Objekten erscheint. Das erreichen Sie beispielsweise folgendermaßen:

- Ziehen Sie einen Rahmen auf, der die Tabelle teilweise oder komplett erfasst.
- Klicken Sie in die Tabelle und betätigen dann die **Esc**-Taste.

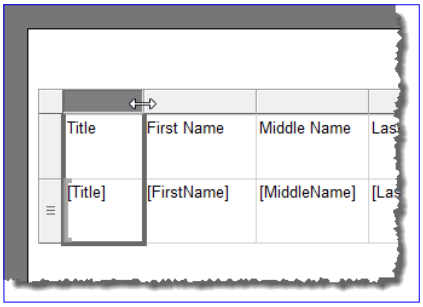


Bild 9: Einstellen der Spaltenbreite

Danach können Sie die Tabelle irgendwo am Rand mit der Maus greifen und an die gewünschte Stelle verschieben.

Größe der Tabelle ändern

Um die Größe einer Tabelle zu ändern, markieren Sie diese wie oben zum Verschieben beschrieben. Danach können Sie die Größe durch Ziehen an einem der Vierecke am Rand der Tabelle einstellen. Die Tabellenfelder passen ihre Höhe und Breite proportional zu den neuen Abmessungen an.

Zeilenhöhe und Spaltenbreite anpassen

Wenn Sie die Zeilenhöhe anpassen wollen, klicken Sie einmal in die Tabelle, damit die grauen Kästchen als Markierung der Spalten und Zeilen erscheinen. Dann können Sie den kleinen grauen Strich zwischen zwei Zeilen zwischen den grauen Kästchen verschieben und so die Höhe der jeweils darüber liegenden Zeile anpassen. Auf die gleiche Weise funktioniert es für die Spalten. Ziehen Sie den Strich zwischen zwei grauen Kästen über den betroffenen Spalten nach links oder rechts, um die Breite der links davon befindliche Spalte einzustellen (siehe Bild 9).

Beachten Sie: Die Breite der rechts von der angepassten Spalte bleibt unverändert. Wenn Sie die Breite vergrößern, werden die rechts davon liegenden Spalten nach rechts verschoben. Gleiches gilt für die Zeilenhöhen.

Spaltenbeschriftungen anpassen

Wenn wir ein Datenmodell in englischer Sprache verwenden, aber eine deutschsprachige Anwendung darauf aufsetzen wollen, können wir mit den englischen Zeilenüberschriften, die automatisch auf Basis der Feldnamen eingefügt wurden, nichts anfangen. Diese können Sie jedoch leicht anpassen. Klicken Sie einmal auf die anzupassende Beschriftung und dann noch einmal oder klicken Sie einmal darauf und geben dann ein Zeichen ein, um den Bearbeitungsmodus zu aktivieren.

Reporting Services: Gruppen und Summen

Im Artikel »Reporting Services: Tabellarische Berichte« haben wir uns am Beispiel einer Kundenliste bereits angesehen, wie Sie einfache Tabellen erstellen und einfache Features wie sich wiederholende Spaltenüberschriften nutzen. In diesem Artikel wollen wir Kunden, Bestellungen und Bestellpositionen zum Thema machen und dazu wiederum eine Tabelle verwenden. Wegen der verschiedenen Ebenen soll diese allerdings Gruppierungen enthalten, die uns unter anderem Informationen über die Umsätze je Position, Bestellung und Kunde liefern.

Vorbereitungen

Als Vorbereitung benötigen Sie die Beispieldatenbank **AdventureWorks**. Deren Download und Installation beschreiben wir im Artikel **AdventureWorks: Schnelle Beispieldatenbank** (**www.datenbankentwickler.net/254**).

Ziel

Wir möchten einen Bericht erstellen, der die Bestellpositionen nach Bestellung und Kunde gruppiert und die Gruppensummen anzeigt. Die Detaildatensätze sollen also jeweils eine Bestellposition enthalten. Die Bestellpositionen einer Bestellung sollen einen Gruppenkopf enthalten, der das Bestelldatum enthält und einen Gruppenfuß mit der Bestellsumme. Außerdem wollen wir noch eine weitere Gruppierung hinzufügen, welche die Bestellungen nach Kunde gruppiert. Diese soll im Gruppenkopf einige wesentliche Kundendaten anzeigen wie die Kundennummer, Vor- und Nachname und die Firma. Im Gruppenfuß dieser Gruppe soll die Summe der Bestellpositionen erscheinen, also der gesamte Umsatz mit diesem Kunden.

Umsatz je Kunde anzeigen

Wir nähern uns dem Schritt für Schritt. Zunächst möchten wir die Umsatzsummen je Kunde in jeweils einer Zeile des Berichts anzeigen. Jede Zeile der Tabelle soll also einen Kunden und die Umsätze dieses Kunden liefern.

Gruppierungen per Assistent

Im ersten Anlauf wollen wir die gruppierte Tabelle per Assistent erstellen. Dazu rufen wir im SQL Server Report Builder mit dem Ribbonbefehl **Datei|Neu** den Dialog **Neuer Bericht oder neues Dataset** auf. Hier wählen wir den Eintrag **Tabellen- oder Matrix-Assistent** (siehe Bild 1).

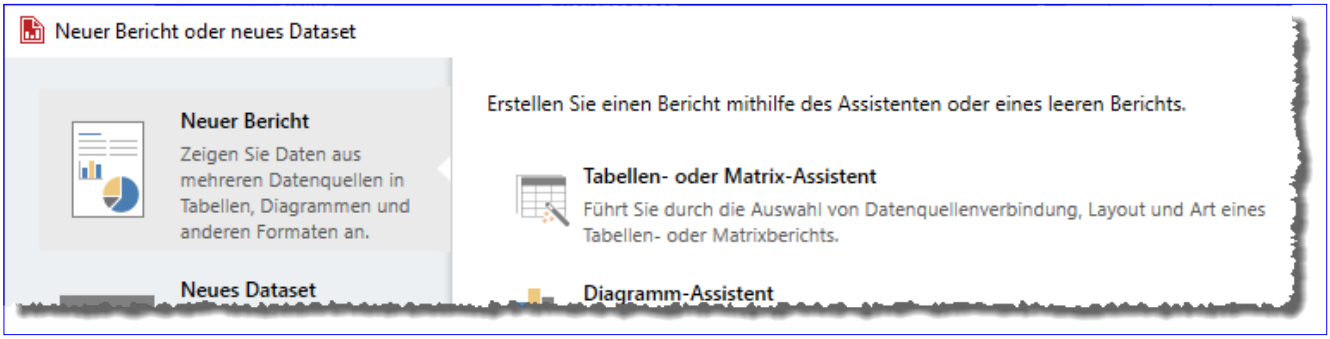


Bild 1: Starten des Assistenten

Datenquelle festlegen

Im nun folgenden Dialog **Neue Tabelle oder Matrix** aktivieren wir die Option **Dataset erstellen** und klicken auf **Weiter**. Damit gelangen wir zum Schritt **Verbindung mit einer Datenquelle auswählen**. Hier klicken Sie auf die Schaltfläche **Neu**. Im Dialog **Datenquelleneigenschaften** definieren Sie die Datenquelle, die beispielsweise wie in Bild 2 aussieht.

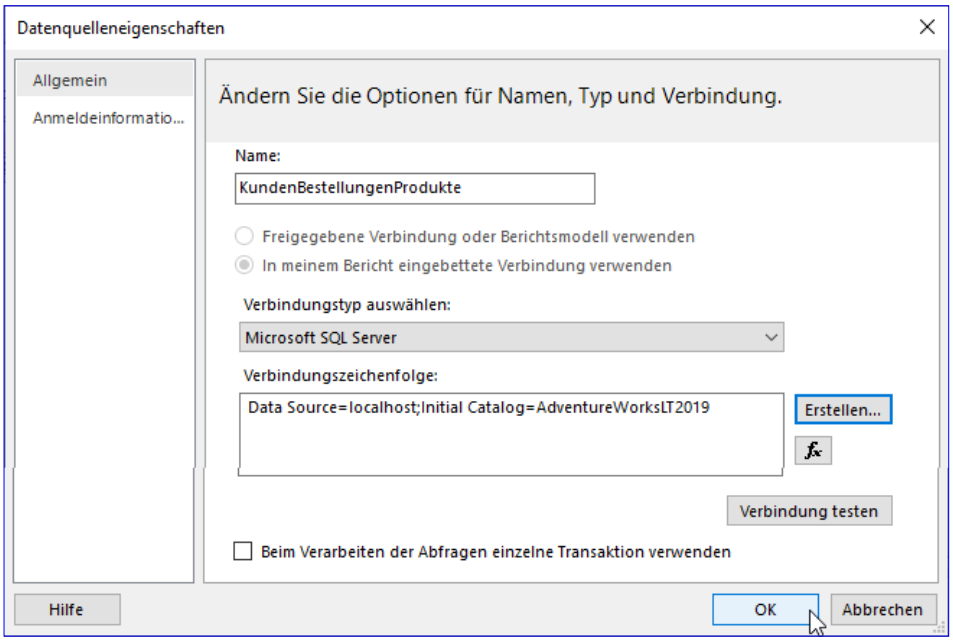


Bild 2: Datenquelle definieren

Dataset definieren

Zurück im Assistenten sehen Sie den Schritt Abfrage entwerfen. Hier wählen Sie im linken Bereich die kompletten Tabellen oder auch nur einzelne Felder aus, die im Bericht verwendet werden sollen.

Aus Gründen der Übersicht sollten Sie nur die Felder wählen, die Sie später auch nutzen. In diesem Fall nutzen wir Felder der Tabellen **Customer**, **Product**, **ProductCategory**, **SalesOrderDetail** und **SalesOrderHeader**. Mit einem Klick auf die Schaltfläche **Abfrage ausführen** können Sie sich die zu erwartenden

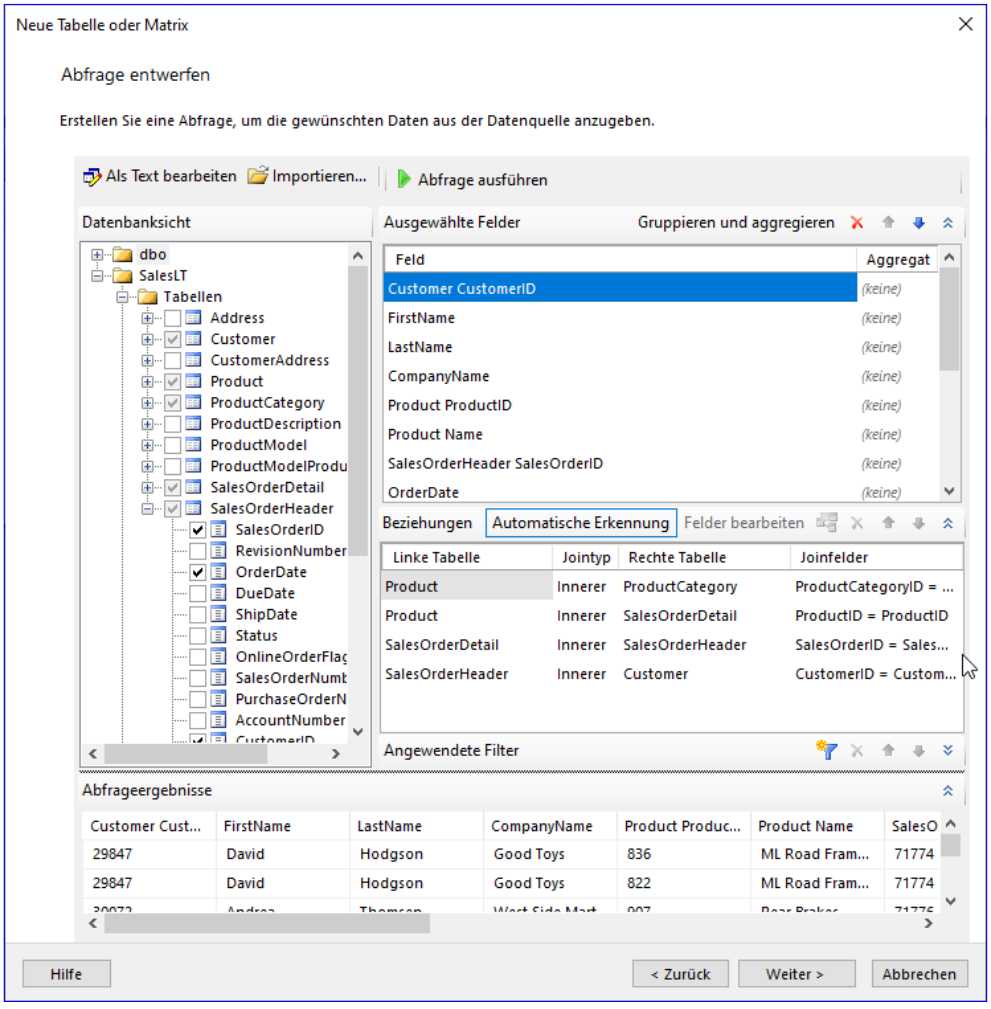


Bild 3: Dataset definieren

Daten direkt als Abfrageergebnis anschauen (siehe Bild 3).

Wie Sie Datenquelle und Dataset ohne Assistent anlegen, beschreiben wir im Detail Artikel **SQL Server Report Builder** (**www.datenbankentwickler.net/258**).

Gruppierungen definieren

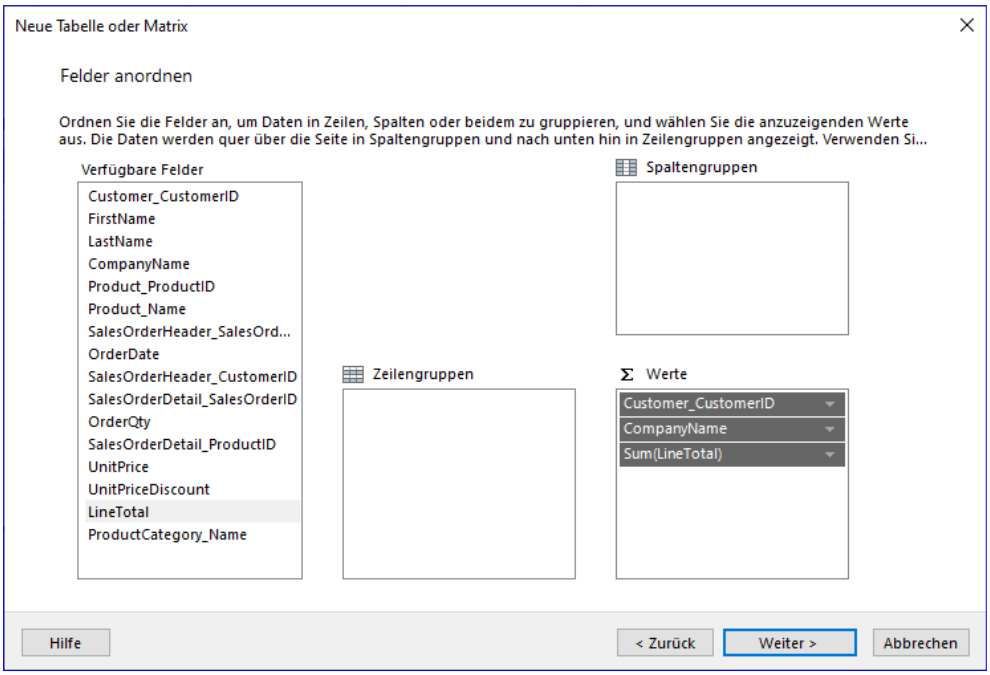


Bild 4: Felder für die Tabelle auswählen

Nachdem wir festgelegt haben, welche Daten wir im Bericht sehen wollen, folgt im nächsten Schritt die Anordnung der Felder. Hier wird es interessant, denn hier finden wir vier Listen vor:

- **Verfügbare Felder**: Alle Felder, die wir im vorherigen Schritt selektiert haben.
- **Spaltengruppen**:
- **Zeilengruppen**:
- **Werte**:

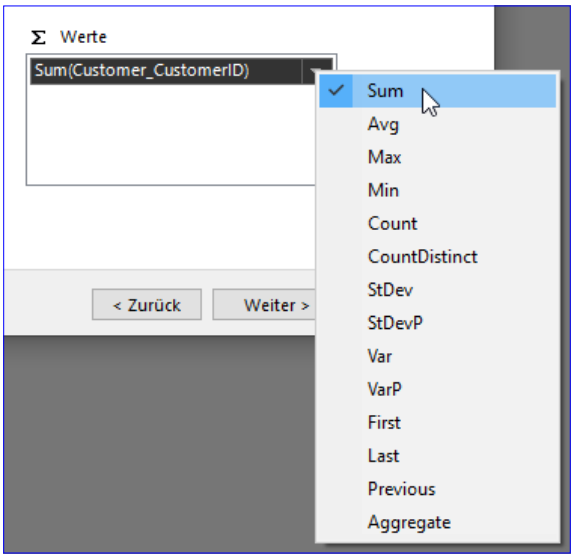


Bild 5: Einstellen der Aggregatfunktion

Im ersten Beispiel wollen wir nur die Umsätze nach Kunden ausgeben. Dazu reicht es aus, wenn wir einfach die anzuzeigenden Felder in den Bereich **Werte** ziehen. Wir wählen hier die Kundennummer (**CustomerID**), die Firmenbezeichnung (**CustomerName**) und den Umsatz je Bestellposition, also das Feld **LineTotal** der Tabelle **SalesOrderDetail** (siehe Bild 4). Für Felder mit numerischen Datentypen stellt der Assistent automatisch die **Sum**-Funktion als Aggregatfunktion ein. Für das Feld **CustomerID** wollen wir das jedoch aus nachvollziehbaren Gründen nicht.

Um dies zu ändern, klicken Sie einfach auf die Schaltfläche mit dem nach unten zeigenden Dreieck. Es erscheint eine Liste aller Aggregatfunktionen. Vielleicht suchen Sie wie in Access-Abfragen nach einem Eintrag wie Ausdruck. Diesen gibt es nicht: Damit einfach nur der Wert des Feldes angezeigt wird, klicken Sie einfach auf die

Zeile, die mit dem Haken markiert ist. Dann ist keine Aggregatfunktion mehr für dieses Feld festgelegt (siehe Bild 5). Wenn Sie das Textfeld **CompanyName** hinzufügen, wird automatisch keine Aggregatfunktion selektiert. Erst für das Feld **LineTotal** legt der Assistent wieder die Aggregatfunktion **Sum** fest, die in diesem Fall ja auch erwünscht ist.

Die folgenden beiden Schritte des Assistenten bieten aufgrund der aktuellen Einstellungen keine weiteren Auswahlmöglichkeiten, sodass wir die Erstellung abschließen können. Das Ergebnis der Arbeit mit dem Assistenten sieht wie in Bild 6 aus. Die Spaltenbreiten haben wir nachträglich vergrößert.

Wenn wir den Bericht nun ausführen und in die **Seitenlayout**-Ansicht wechseln, erhalten wir den Bericht mit dem Ausschnitt aus Bild 7. Die angezeigten Daten entsprechen unseren Wünschen, allein am Layout könnte man noch arbeiten. Beispielsweise sollen die Spaltenüberschriften deutsche Texte erhalten und der Umsatz soll in der Währung **EUR** angezeigt werden.

Ebene für die einzelnen Bestellpositionen einziehen

Nun wollen wir ausgehend von einem Bericht, der alle Bestellpositionen anzeigt, mit weiteren Gruppierungen arbeiten, die auch Kopf- und Fußbereiche enthalten. Wir starten mit einem neuen, leeren Bericht, für den wir wieder eine Datenquelle auf Basis der Datenbank **AdventureWorks** verwenden und ein Dataset mit den fünf oben verwendeten Tabellen. Dem Bericht fügen wir eine neues Tabellensteuerelement (Tablix) hinzu. Die Überschrift und das Textfeld im Fußbereich entfernen wir, ebenso den Fußbereich.

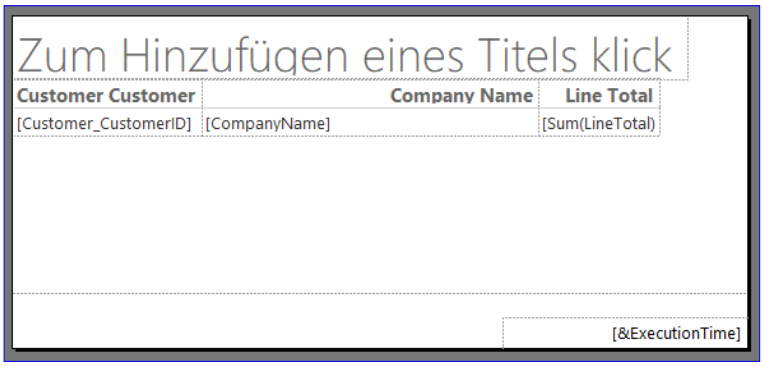
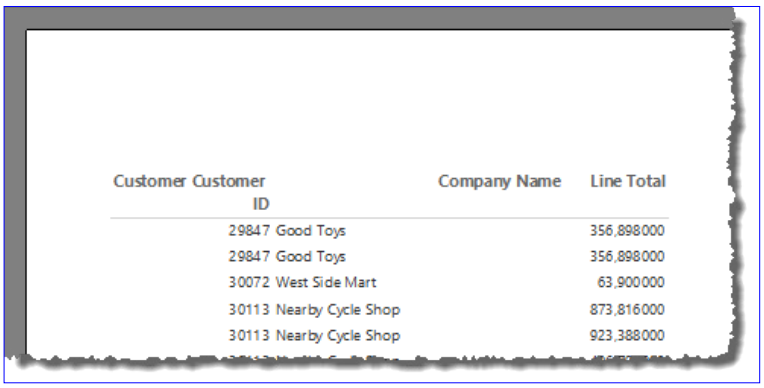


Bild 6: Resultierender Entwurf des Assistenten



Customer Customer ID	Company Name	Line Total
29847	Good Toys	356,898000
29847	Good Toys	356,898000
30072	West Side Mart	63,900000
30113	Nearby Cycle Shop	873,816000
30113	Nearby Cycle Shop	923,388000

Bild 7: **Seitenlayout**-Ansicht des Berichts

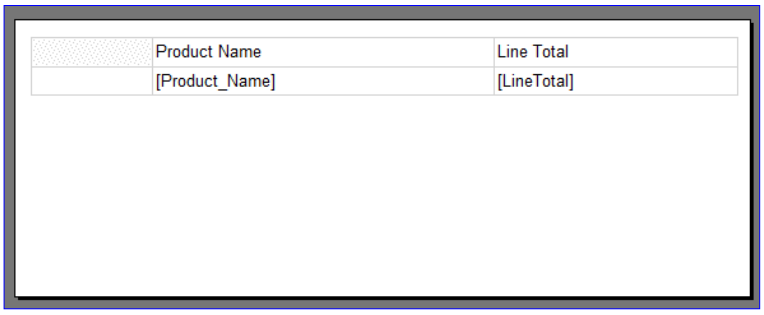
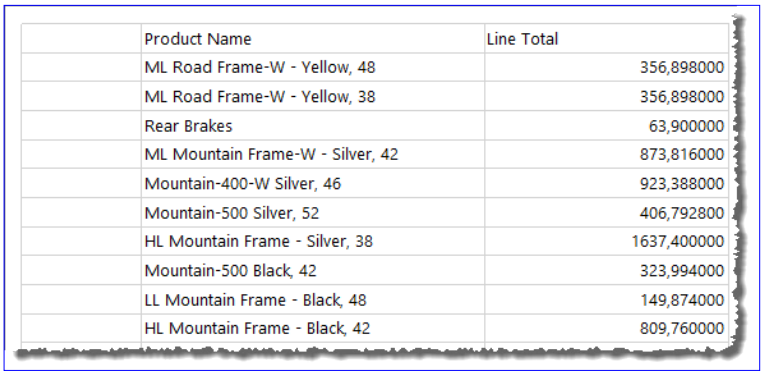


Bild 8: Entwurf des Berichts zur Anzeige der Artikel



Product Name	Line Total
ML Road Frame-W - Yellow, 48	356,898000
ML Road Frame-W - Yellow, 38	356,898000
Rear Brakes	63,900000
ML Mountain Frame-W - Silver, 42	873,816000
Mountain-400-W Silver, 46	923,388000
Mountain-500 Silver, 52	406,792800
HL Mountain Frame - Silver, 38	1637,400000
Mountain-500 Black, 42	323,994000
LL Mountain Frame - Black, 48	149,874000
HL Mountain Frame - Black, 42	809,760000

Bild 9: **Seitenlayout**-Ansicht des Berichts zur Anzeige der Artikel