

# TSQL Mod – Eine DLL für schnelle SQL Server arbeiten

## TSQLmod Download

### Klasse db (Connection String)

benötigt in erster Linie die SQL Instanz. Nachdem das Objekt erfolgreich initialisiert wurde, wird auch gleichzeitig die Verbindung aufgemacht und die folgenden Methoden können genutzt werden.

#### LookUP(...)

gibt aus einem SQL Query den ersten Treffer der angegebenen Spalte als String wieder. Ideal um einen Wert aus der Datenbank auszulesen. Möglich ist es entweder die Spaltennummer oder den Spaltennamen anzugeben.

#### getRowList(...)

gibt eine List<string> oder generische List<T> von der angegebenen Spalte zurück. Man erhält quasi aus dem Select eine gewünschte Spalte

#### getRowStringBuilder(...)

ähnlich wie die getRowList(...) ist der Rückgabewert aber ein StringBuilder, in welchen alle Zeilen einer selektierten Spalte enthalten sind.

#### getDynamicList(...)

erfordert eine Klasse welche dieselben Datentypen und Bezeichnung hat wie das SQL Select. Als Rückgabe erhält man man eine List<meineKlasse>, welche 1:1 so viele Elemente und Spalten hat wie das Sql Query. Das ganze arbeitet nicht

mit Reflektionen, sondern nach dem Prinzip von diesem genialen Autor: KLICK

## Dazu jeweils ein Beispiel. Ausgehend vom folgenden Select:

```
SELECT TOP 20 [ContactTypeID]
      , [Name]
      , [ModifiedDate]
FROM [AdventureWorks2014].[Person].[ContactType]
```

100 %

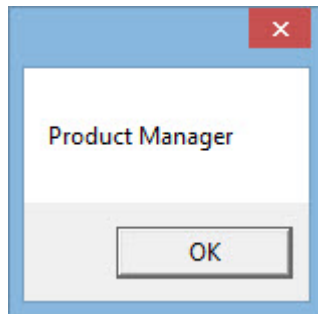
Ergebnisse    Meldungen

|    | ContactTypeID | Name                            | ModifiedDate            |
|----|---------------|---------------------------------|-------------------------|
| 1  | 1             | Accounting Manager              | 2008-04-30 00:00:00.000 |
| 2  | 2             | Assistant Sales Agent           | 2008-04-30 00:00:00.000 |
| 3  | 3             | Assistant Sales Representative  | 2008-04-30 00:00:00.000 |
| 4  | 4             | Coordinator Foreign Markets     | 2008-04-30 00:00:00.000 |
| 5  | 5             | Export Administrator            | 2008-04-30 00:00:00.000 |
| 6  | 6             | International Marketing Manager | 2008-04-30 00:00:00.000 |
| 7  | 7             | Marketing Assistant             | 2008-04-30 00:00:00.000 |
| 8  | 8             | Marketing Manager               | 2008-04-30 00:00:00.000 |
| 9  | 9             | Marketing Representative        | 2008-04-30 00:00:00.000 |
| 10 | 10            | Order Administrator             | 2008-04-30 00:00:00.000 |
| 11 | 11            | Owner                           | 2008-04-30 00:00:00.000 |
| 12 | 12            | Owner/Marketing Assistant       | 2008-04-30 00:00:00.000 |
| 13 | 13            | Product Manager                 | 2008-04-30 00:00:00.000 |
| 14 | 14            | Purchasing Agent                | 2008-04-30 00:00:00.000 |
| 15 | 15            | Purchasing Manager              | 2008-04-30 00:00:00.000 |
| 16 | 16            | Regional Account Representative | 2008-04-30 00:00:00.000 |
| 17 | 17            | Sales Agent                     | 2008-04-30 00:00:00.000 |
| 18 | 18            | Sales Associate                 | 2008-04-30 00:00:00.000 |
| 19 | 19            | Sales Manager                   | 2008-04-30 00:00:00.000 |
| 20 | 20            | Sales Representative            | 2008-04-30 00:00:00.000 |

einem erstellten Objekt der Klasse db:  
[crayon-67678ffa5e14e097116644/]  
und ein string mit folgendem select:  
[crayon-67678ffa5e153055599043/]

## LookUp (...)

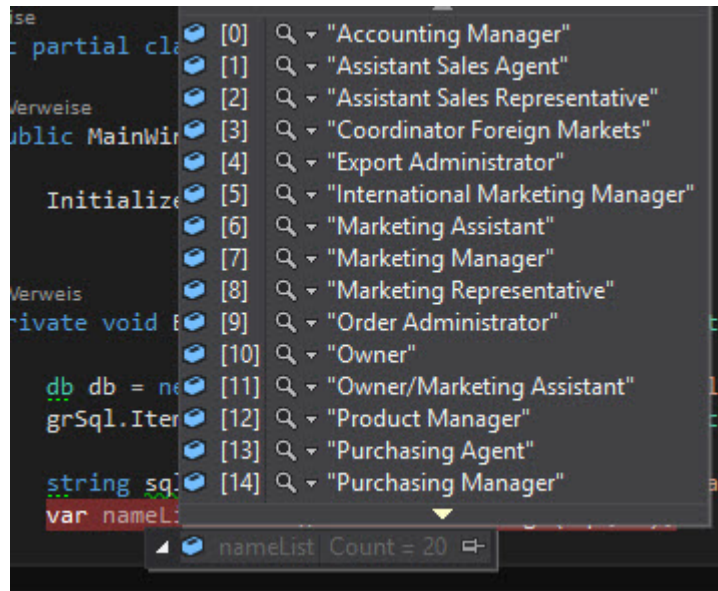
[crayon-67678ffa5e155514193188/]



Antwort:

## getRowList (...)

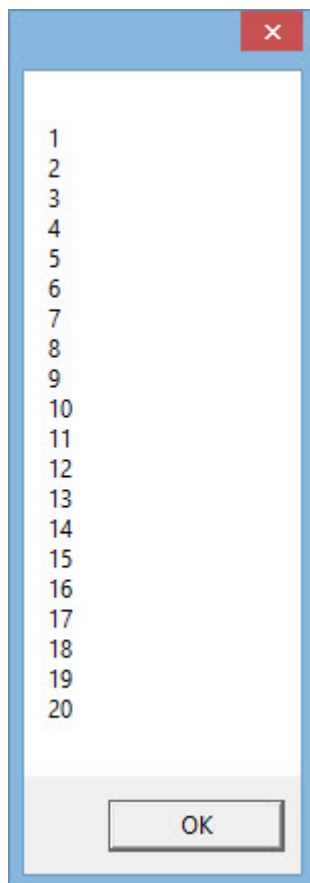
[crayon-67678ffa5e16f980886300/]



Antwort:

**getRowStringBuilder(...)**

[crayon-67678ffa5e171427280887/]



Antwort:

**getDynamicList(...)**

Wie oben bereits erwähnt, ist hierfür eine Klasse mit

Property's notwendig. Diese kann man ganz einfach auch mit den Methoden aus `CreateClass` – Klasse erstellen. Dazu weiter unten.

```
[crayon-67678ffa5e172804911427/]
```

dann kann man so eine dynamische Liste ganz einfach erstellen:

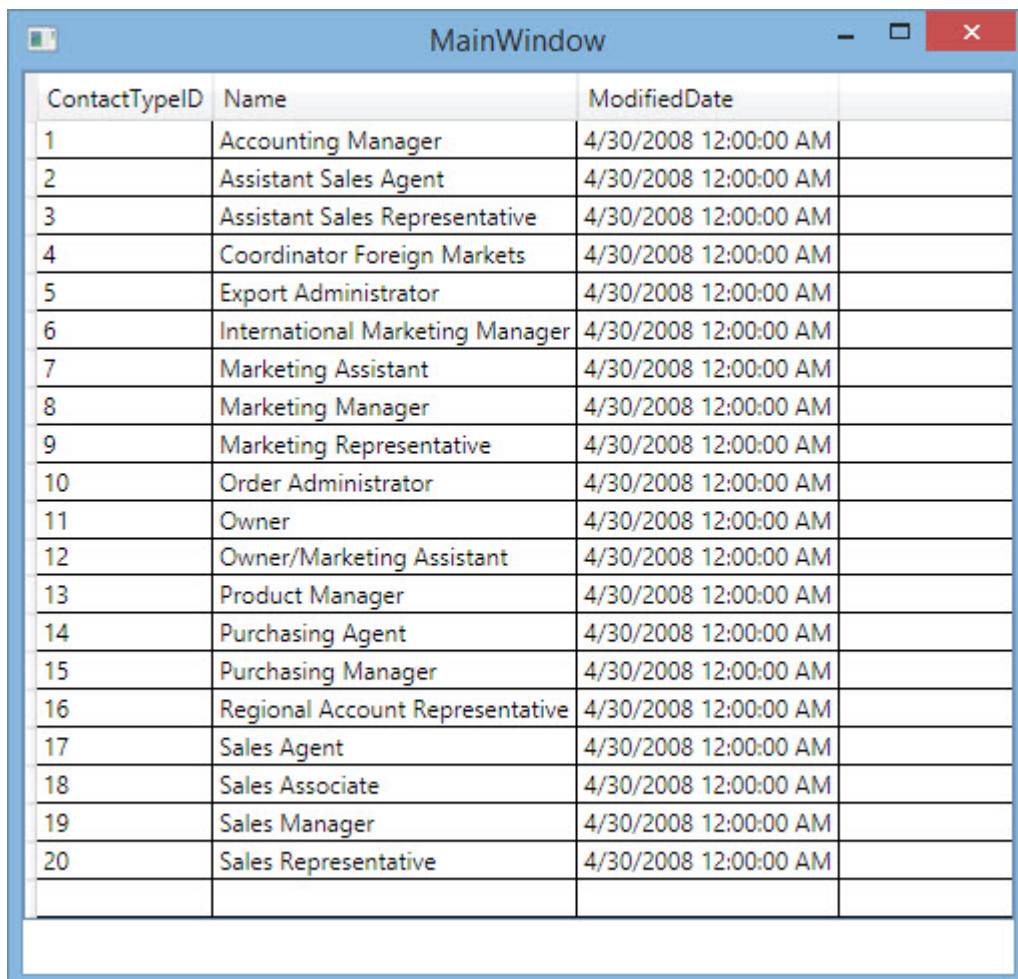
```
[crayon-67678ffa5e173254053872/]
```

diese Liste kann man nun z.B. einem Datagrid aus WPF zuordnen:

```
[crayon-67678ffa5e175489094784/]
```

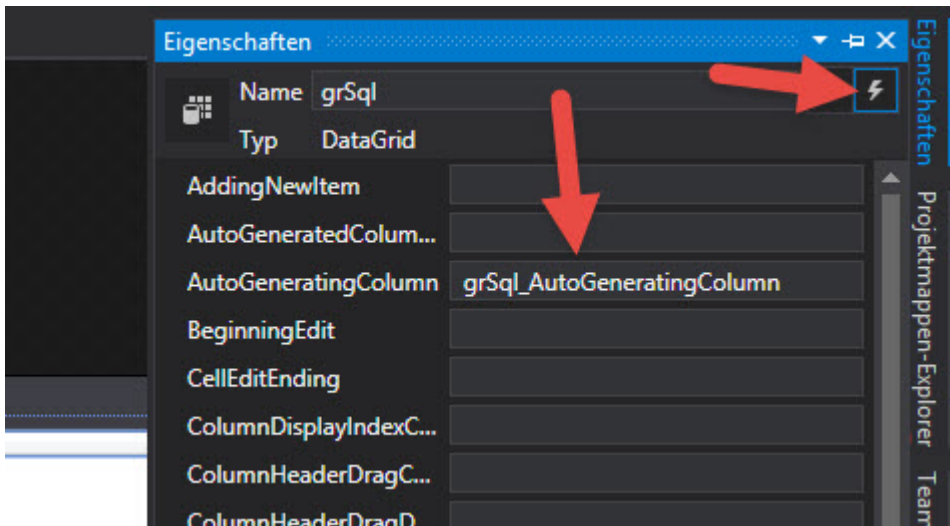
```
[crayon-67678ffa5e176348823657/]
```

Das ganze sieht dann so aus:



| ContactTypeID | Name                            | ModifiedDate          |  |
|---------------|---------------------------------|-----------------------|--|
| 1             | Accounting Manager              | 4/30/2008 12:00:00 AM |  |
| 2             | Assistant Sales Agent           | 4/30/2008 12:00:00 AM |  |
| 3             | Assistant Sales Representative  | 4/30/2008 12:00:00 AM |  |
| 4             | Coordinator Foreign Markets     | 4/30/2008 12:00:00 AM |  |
| 5             | Export Administrator            | 4/30/2008 12:00:00 AM |  |
| 6             | International Marketing Manager | 4/30/2008 12:00:00 AM |  |
| 7             | Marketing Assistant             | 4/30/2008 12:00:00 AM |  |
| 8             | Marketing Manager               | 4/30/2008 12:00:00 AM |  |
| 9             | Marketing Representative        | 4/30/2008 12:00:00 AM |  |
| 10            | Order Administrator             | 4/30/2008 12:00:00 AM |  |
| 11            | Owner                           | 4/30/2008 12:00:00 AM |  |
| 12            | Owner/Marketing Assistant       | 4/30/2008 12:00:00 AM |  |
| 13            | Product Manager                 | 4/30/2008 12:00:00 AM |  |
| 14            | Purchasing Agent                | 4/30/2008 12:00:00 AM |  |
| 15            | Purchasing Manager              | 4/30/2008 12:00:00 AM |  |
| 16            | Regional Account Representative | 4/30/2008 12:00:00 AM |  |
| 17            | Sales Agent                     | 4/30/2008 12:00:00 AM |  |
| 18            | Sales Associate                 | 4/30/2008 12:00:00 AM |  |
| 19            | Sales Manager                   | 4/30/2008 12:00:00 AM |  |
| 20            | Sales Representative            | 4/30/2008 12:00:00 AM |  |
|               |                                 |                       |  |

wem das Datumsformat stört, der kann dem Ereignis `AutoGeneratingColumn` aus dem `DataGrid` eine Änderung des Datumsformates durchführen:



folgendes soll nun passieren, wenn das Ereignis eintrifft:  
[crayon-67678ffa5e177870599614/]  
Nun sieht das ganze so aus:

| ContactTypeID | Name                            | ModifiedDate |
|---------------|---------------------------------|--------------|
| 1             | Accounting Manager              | 30.04.2008   |
| 2             | Assistant Sales Agent           | 30.04.2008   |
| 3             | Assistant Sales Representative  | 30.04.2008   |
| 4             | Coordinator Foreign Markets     | 30.04.2008   |
| 5             | Export Administrator            | 30.04.2008   |
| 6             | International Marketing Manager | 30.04.2008   |
| 7             | Marketing Assistant             | 30.04.2008   |
| 8             | Marketing Manager               | 30.04.2008   |
| 9             | Marketing Representative        | 30.04.2008   |
| 10            | Order Administrator             | 30.04.2008   |
| 11            | Owner                           | 30.04.2008   |
| 12            | Owner/Marketing Assistant       | 30.04.2008   |
| 13            | Product Manager                 | 30.04.2008   |
| 14            | Purchasing Agent                | 30.04.2008   |
| 15            | Purchasing Manager              | 30.04.2008   |
| 16            | Regional Account Representative | 30.04.2008   |
| 17            | Sales Agent                     | 30.04.2008   |
| 18            | Sales Associate                 | 30.04.2008   |
| 19            | Sales Manager                   | 30.04.2008   |
| 20            | Sales Representative            | 30.04.2008   |

---

# generische Listen / Collection

Zwar braucht die Dictionary<T, K> weniger Zeit beim hinzufügen/ löschen von Werten, braucht die SortedList doch weniger Ramkapazitäten und ist im großen und ganzen schneller. Deutlich langsamer ist die Hashtable und SortedDictionary.

<T> = Typenparameter, erwartet wird ein Datentyp als Parameter. Beispiel string, int, object,...

<V, K> = siehe <T>, jedoch wegen besserer Lesbarkeit stellt dieser Typenparameter den Wert des Value bzw. Key da.

## List<T>

[crayon-67678ffa5e609326355385/]

## SortedList<V, K>

[crayon-67678ffa5e60d174407512/]

Beinhaltet einen Key und Value. Über den Key lässt sich das Value herausfinden. Beispiel:

[crayon-67678ffa5e60f470267253/]

liefert den Wert Boolean Wert True

## Dictionary<V, K>

[crayon-67678ffa5e610573700764/]

## ArrayList (Object)

[crayon-67678ffa5e611654210501/]

## Hashtable (Object K, Object V)

[crayon-67678ffa5e612547953644/]

## ObservableCollection

[crayon-67678ffa5e613877846594/]

Im Gegensatz zur `List<T>` nutzt die `ObservableCollection` die `INotifyCollectionChanged` Schnittstelle. Diese gibt eine Meldung, sobald sich in der `Collection` etwas geändert hat. Sehr Sinnvoll, wenn man diese `Collection` an ein Steuerelement per WPF binden möchte, aktualisiert sich das Element so automatisch. Ein Nachteil ist jedoch, dass man die Liste nicht aus der Liste suchen/sortieren kann. Hier kann man nachlesen, wie man dies doch mit einbauen kann -> [Link](#)

## **List<Tuble<T,A,B>**

Bisher hatten wir immer nur die Möglichkeit über `Key / Value` ein Pärchen vom Eintrag zu bilden. Manchmal kommt man aber in die Situation wo man nicht das `Key/Value` Prinzip haben möchte, oder mehr als 2 Argumente übergeben möchte. Da kommt das seit .NET 4.0 eingeführte `Tuble` ins Spiel. Die einzelnen Einträge werden dann als `Item1`, `Item2` usw. innerhalb des Eintrags geführt.

[crayon-67678ffa5e614295005423/]

*Quelle: <http://blog.bodurov.com/Performance-SortedList-SortedDictionary-Dictionary-Hashtable/>*

---

# **generische Klassen**

Mit generischen Klassen kann man Datentyp unabhängigen Aufbau einer Klasse erreichen. Das bedeutet, man kann dann ein Objekt der Klasse erzeugen und sagen, dass die Methoden dort eben mit diesen Datentyp arbeiten soll, dem wir dem Objekt übergeben. Auch eine dort deklarierte Variable ist eben der Typ, der übergeben wurde.

Dabei arbeitet man mit einem sogenannten `Typ-Parameter`,



welcher ähnlich einem Platzhalter für einen Datentyp stehen soll. Deutlicher wird das vielleicht durch das folgende Beispiel:

```
[crayon-67678ffa5e880660680641/]
```

Die Klasse erhält einen Typenparameter T. Man könnte auch einen anderen Buchstaben nehmen (i.d.R. ist der erste immer ein T), oder durch Komma getrennt weitere Datentypen anfügen.

Etwas ungewöhnlicher sieht da die Klassen-Eigenschaft in der 2. Zeile aus. Denkt man sich das T weg, könnte dort ein int, string[], double oder sonstwas stehen.

Dann folgt ein Konstruktor, der dieser Eigenschaft nun einen Wert zuweist

und zum Schluss noch eine Methode, die diese Eigenschaft ausgibt.

### **Constraints**

im oberen Beispiel hätte man rein theoretisch die Möglichkeit alle möglichen Datentypen zu setzen. Möchte man dies aber auf eine bestimmte Art von Datentypen beschränken, so gibt es eine where Klausel. Das Muster erinnert dann ein wenig an SQL.

Dann nimmt die Klasse folgende Bezeichnung ein:

```
[crayon-67678ffa5e885587222005/]
```

Dabei kann man daraus den Satz machen: „Filtere alle, die von der Elternklasse IComparable erben“. Dabei macht man sich am besten im Objektexplorer schlau, welche Einschränkungen man treffen möchte.

Möchte man ein Objekt erstellen und dieser Datentyp passt nicht unter das Gefilterte, wird bereits zur Entwicklungszeit ein Fehler angezeigt.

Quelle: <http://www.dotnetperls.com/generic>