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 %

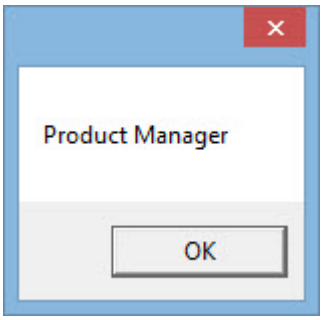
Ergebnisse | Meldungen

	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-600bd9283b141547440322/]
und ein string mit folgendem select:
[crayon-600bd9283b145328266636/]

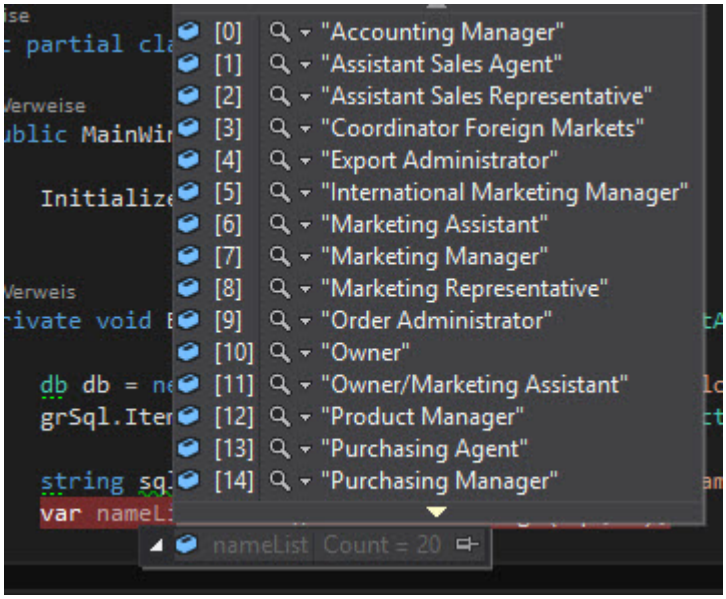
LookUp(…)

[crayon-600bd9283b147758599005/]

Antwort: 

getRowList(…)

[crayon-600bd9283b148088556854/]

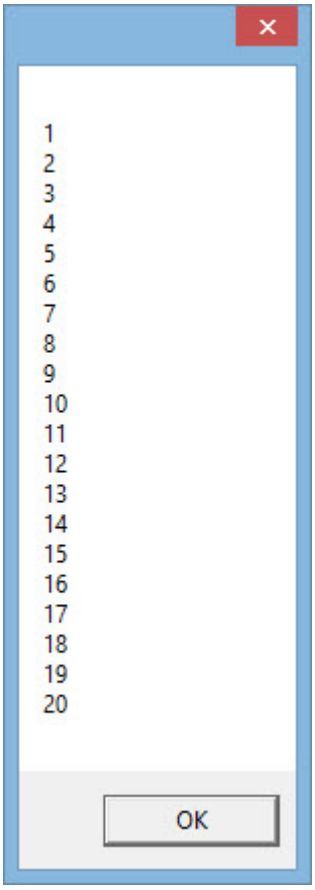


Antwort:

getRowStringBuilder(…)

[crayon-600bd9283b149702350400/]

Antwort:

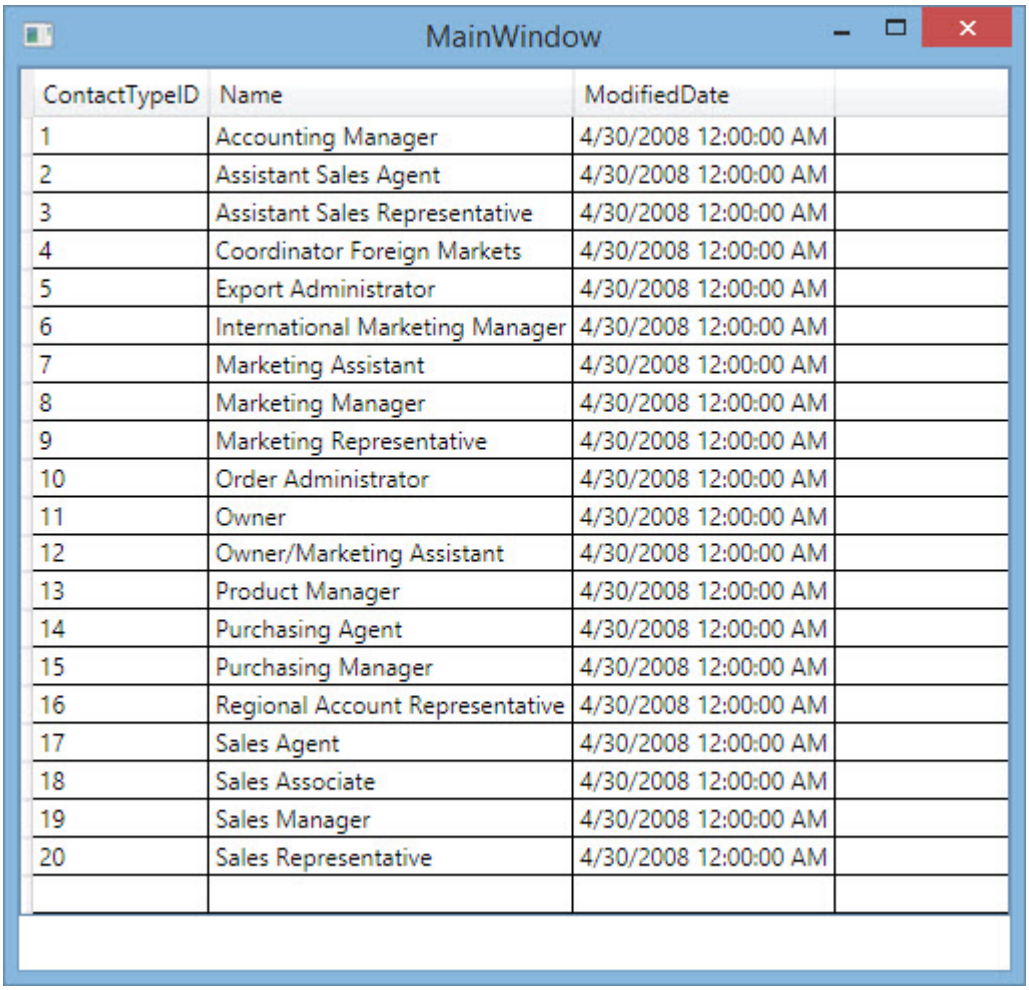


getDynamicList(…)

Wie oben bereits erwähnt, ist hierfür eine Klasse mit

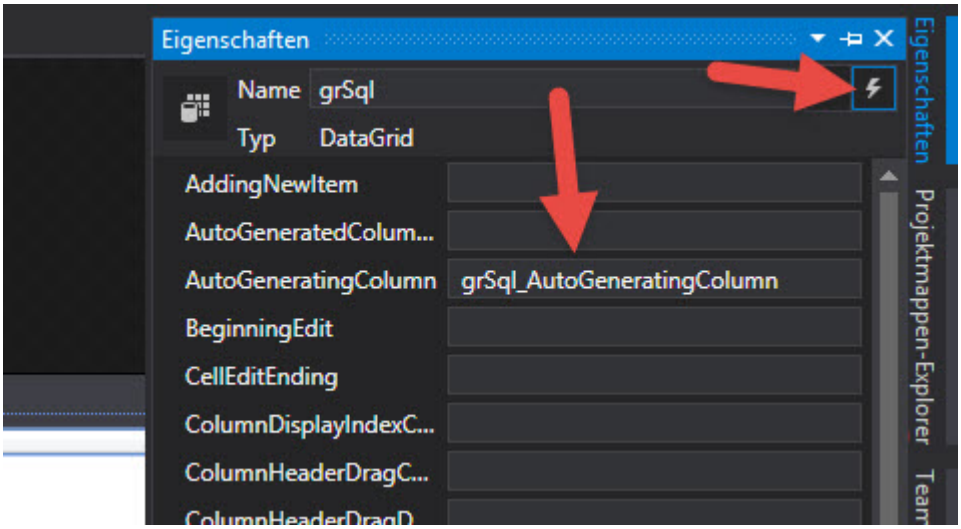
Propertys notwendig. Diese kann man ganz einfach auch mit den Methoden aus CreateClass – Klasse erstellen. Dazu weiter unten.
[crayon-600bd9283b14b813608429/]
dann kann man so eine dynamische Liste ganz einfach erstellen:
[crayon-600bd9283b14c358834510/]

diese Liste kann man nun z.B. einem Datagrid aus WPF zuordnen:
[crayon-600bd9283b14d373821422/]
[crayon-600bd9283b14e341363545/]
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-600bd9283b14f930068999/]
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 **V**alue bzw. **K**ey da.

List<T>
[crayon-600bd9283b8ae110533233/]
SortedList<V, K>
[crayon-600bd9283b8b1485817230/]
Beinhaltet einen Key und Value. Über den Key lässt sich das Value herausfinden. Beispiel:
[crayon-600bd9283b8b2682905748/]
liefert den Wert Boolean Wert True

Dictionary<V, K>
[crayon-600bd9283b8b3438621701/]
ArrayList (Object)
[crayon-600bd9283b8b4180236448/]
Hashtable (Object K, Object V)
[crayon-600bd9283b8b5338860519/]
ObservableCollection
[crayon-600bd9283b8b6330499410/]

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-600bd9283b8b7635477497/]

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 **T**yp-Parameter,

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

[crayon-600bd9283ba6a261434209/]
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.

Contraints

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-600bd9283ba6d148600888/]
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