Binding Checkbox an eine Visible Eigenschaft eines Elements

Ab .NET Framework 4.5 bietet Microsoft den BooleanToVisibilityConverter an mit dessen Hilfe man unter WPF man es umsetzen kann, dass eine Checkbox ein Element Visible oder Hidden setzt.
[crayon-67501c450ee48434861573/]
Quelle: http://stackoverflow.com/questions/6990982/wpf-control-tabitem-visibility-from-a-checkbox

---

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

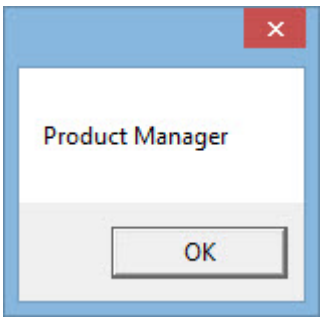
	ContactTypeID	Name	ModifiedDate
1	1	Accounting Manager	2008-04-30 00:00:00.000
2	2	Assistant Sales Agent	2008-04-30 00:00:00.000
3	3	Assistant Sales Representative	2008-04-30 00:00:00.000
4	4	Coordinator Foreign Markets	2008-04-30 00:00:00.000
5	5	Export Administrator	2008-04-30 00:00:00.000
6	6	International 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-67501c450f410411986872/]
und ein string mit folgendem select:
[crayon-67501c450f415325303649/]

LookUp(…)

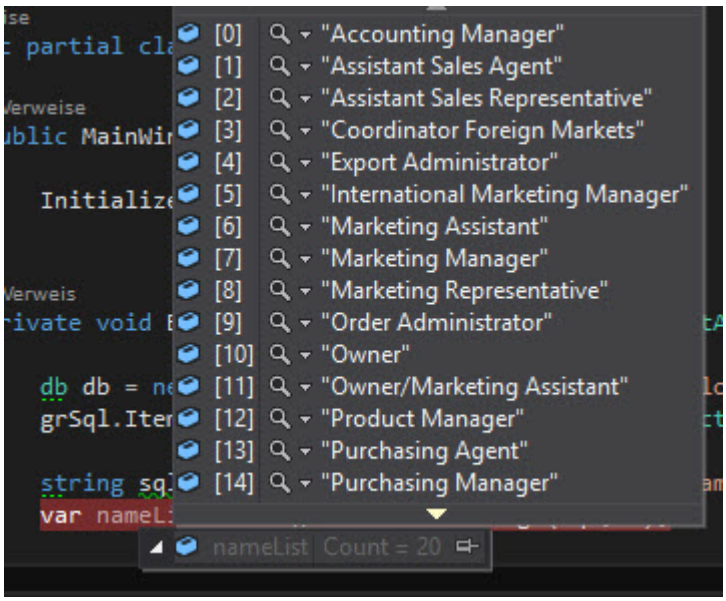
[crayon-67501c450f417406128429/]

Antwort:



getRowList(…)

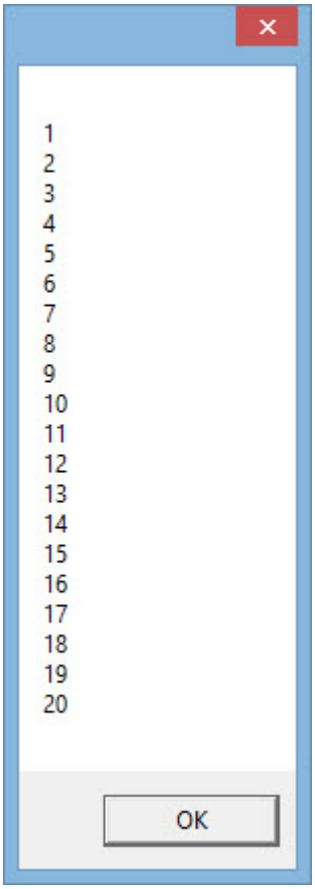
[crayon-67501c450f419903141067/]



Antwort:

getRowStringBuilder(…)

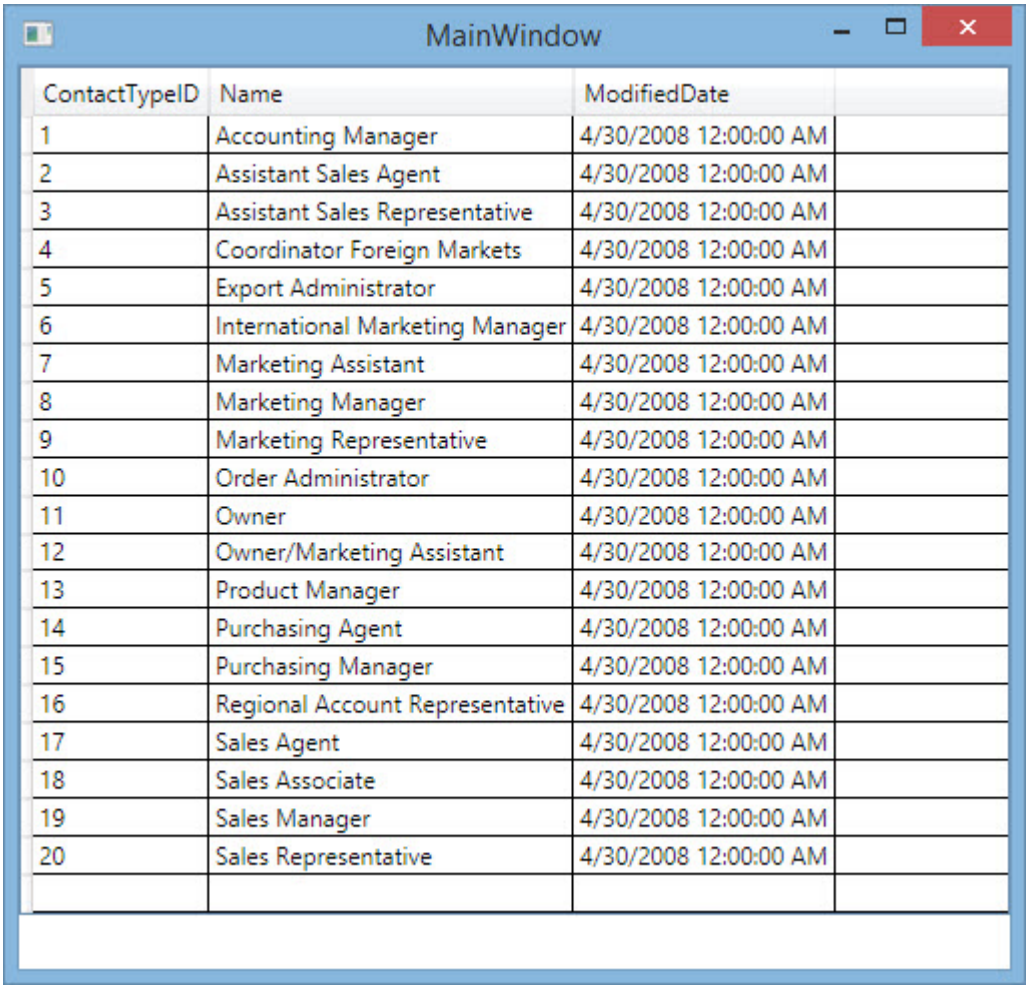
[crayon-67501c450f41a045810267/]

Antwort: 

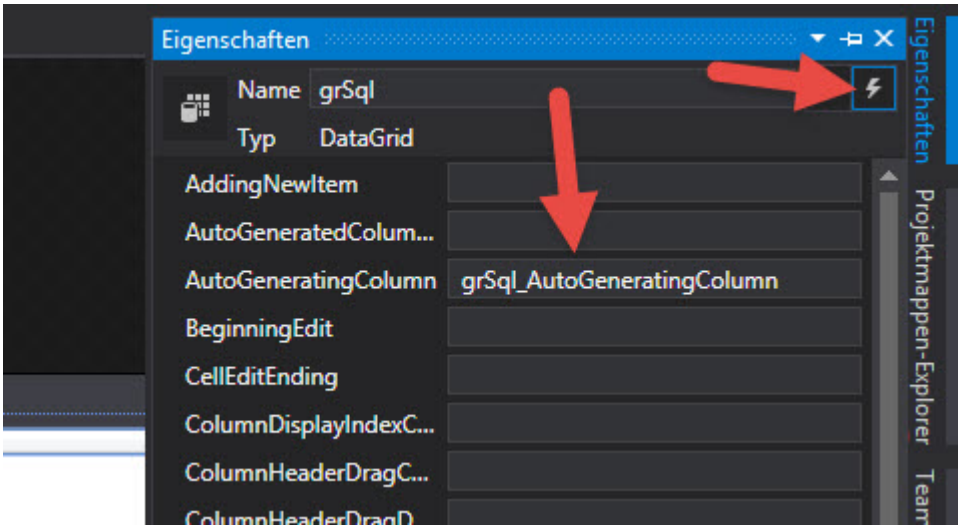
getDynamicList(…)

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-67501c450f41c020200668/]
dann kann man so eine dynamische Liste ganz einfach erstellen:
[crayon-67501c450f41e429109170/]

diese Liste kann man nun z.B. einem Datagrid aus WPF zuordnen:
[crayon-67501c450f41f228984224/]
[crayon-67501c450f421512212260/]
Das ganze sieht dann so aus:



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-67501c450f422924282619/]
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

Button Color aus Hexwert ändern

[crayon-67501c450f671024072255/]
um Button zurück zu setzen:
[crayon-67501c450f675063725153/]

guter einfacher ColorPicker:

http://www.colorpicker.com/

Aus einer anderen Klasse, aus einem anderen Thread in MainWindow schreiben

Möchte man aus einer anderen Klasse, aus einem anderen Thread etwas in die Mainklasse Steuerelemente schreiben, stößt man auf 2 Probleme:

1. Man kann aus Thread 2 nicht in Thread 1 schreiben

2. Man kann nicht, ohne ein Objekt angelegt zu haben nicht in die Steuerelemente schreiben.

Abhilfe schafft ein kleiner Trick.

MainWindow.cs:
[crayon-67501c450f824654669875/]
[crayon-67501c450f827286901624/]

meineAndereKlasse.cs:
[crayon-67501c450f829283144245/]
Kleine Erweiterung, selbes Prinzip um ein Imagecontrol zu ändern:

MainWindow.cs:
[crayon-67501c450f82a903683593/]
meineAndereKlasse.cs:
[crayon-67501c450f82b669732615/]

Quelle: *Stackoverflow*

Schnell mal ein Bild einfügen

Dazu einfach ein Bild in den Designer ziehen.

Möchte man im Code-Behind, dann das Bild ersetzen, gibt man der Image-Control einen Namen und führt folgendes aus:
[crayon-67501c450fb36516366095/]

Aktuelles Datum binden

als namespace hinzufügen:
[crayon-67501c450fe09007558430/]
Den DateTimePicker erstellen:
[crayon-67501c450fe0f241967694/]

Binding Elementbinding

Möchte man eine Elementeigenschaft an die andere binden, muss man in die Quelleigenschaft gehen und ein {Binding … } einsetzen.

Zuerst wird das Objekt, dann dessen Eigenschaft abgefragt.

ElementName (welches Element?)

Path(welche Eigenschaft?)

Bei Path könnte auch etwas anderes stehen. SelectedIndex z.B.

oder Items[0] für das erste Item in der ListBox
[crayon-67501c45100d0179631728/]
Das Visual Studio bietet mit dem Intellisense eine große Hilfe bei der Selektion.

Binding Allgemein

Unter Binding versteht man, eine Eigenschaft von dem Quellobjekt an eine Eigenschaft des Zielobjekts bindet.

Dabei gibt es 2 Arten:

Element an ein anderes Element / Element an Datenquelle z.B. Datenbank

Dabei bindet man eine Abängigkeits-Eigenschaft (Dependency Property) an die andere. Diese müssen gleichen Typs sein. Sind diese Anforderungen erfüllt, kann man alles binden. Breite eines Steuerelements an die Fensterbreite, selektiertes Element einer Combobox an ein Label und sogar ganze Spalte aus einer Datenbank.

Das Visual Studio bietet hierfür eine Hilfe in den Eigenschaften der Steuerelemente an.

Hier sehen wir nun, 7 Bindungstypen, die uns zur Verfügung stehen. Jedes dieser Bindungen umfasst eine Ansammlung von Eigenschaften aus seinem Gebiet.

1. **Datenkontext**: Jede WPF Anwendung hat zu Beginn 2 Schichten. Eine für UI (User Interface) und die andere für Daten. Die Datenschicht ist zu Beginn Null, das heißt ohne Inhalt. Indem wir deren Eigenschaft DataContext setzen umfasst dieser Kontext nun alle angegebenen Daten. Solange man nichts weiter angibt, erben nun die Kindobjekte (Steuerelemente wie Label, Textbox usw.) von dem Elternobjekten diese Daten.
2. **Datenquelle**:
3. **ElementName:** Umfasst alle visuelle und nicht visuelle Elemente aus dem XAML Projekt. Hier stehen einem sämtliche Eigenschaften der Steuerelemente wie Textbox, Label, Combobox und auch nicht die Eigenschaften vom Window zur Verfügung.
4. **RelativeSource FindAncestor:** Hier werden sämtliche Eigenschaften von allen hierarchisch überliegenden Objekten zur Verfügung gestellt.
5. **RelativeSource PreviusData:** Gegenteil vom FindAncestor, werden hier alle unterliegenden Objekteigenschaften aufgelistet
6. **RelativeSource Self:** Hier bekommt man Eigenschaften vom selben Objekt zur Verfügung gestellt
7. **RelativeSource TemplatedParent:**

8. **StaticResource**: Bezeichnet eine schon zuvor in die Resources eingeschriebene Datenquelle