TMS FNC Databinding
DEVELOPERS GUIDE

January 2022

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

Index

Getting started

Included in the TMS FNC Core is a databinding component (TTMSFNCDataBinder) that is capable of binding multiple components to datasets. In combination with a designtime & runtime editor, you can easily manage and update new and existing bindings …

To get started with the databinding component, drop a TTMSFNCDataBinder (further referred to as databinder) component on the form. The databinder component is capable of binding single value components, lists, tables and grids as well as non published properties such as generic TList & TObjectList (Integer or String based). Note that the databinder component is currently only supporting retrieval and display of data. It also automatically updates when a dataset change occurs.

The chapters below demonstrate what is possible based on a TClientDataSet connected to the biolife.xml file provided by Embarcadero, a TDataSource and a single instance of TTMSFNCDataBinder.

Binding a single value

Binding a single value can be done by adding an item to the databinder component, specifying the component, the field name and the property name. It shows the value based on the active record. There are several ways that this can be done.

Programmatically

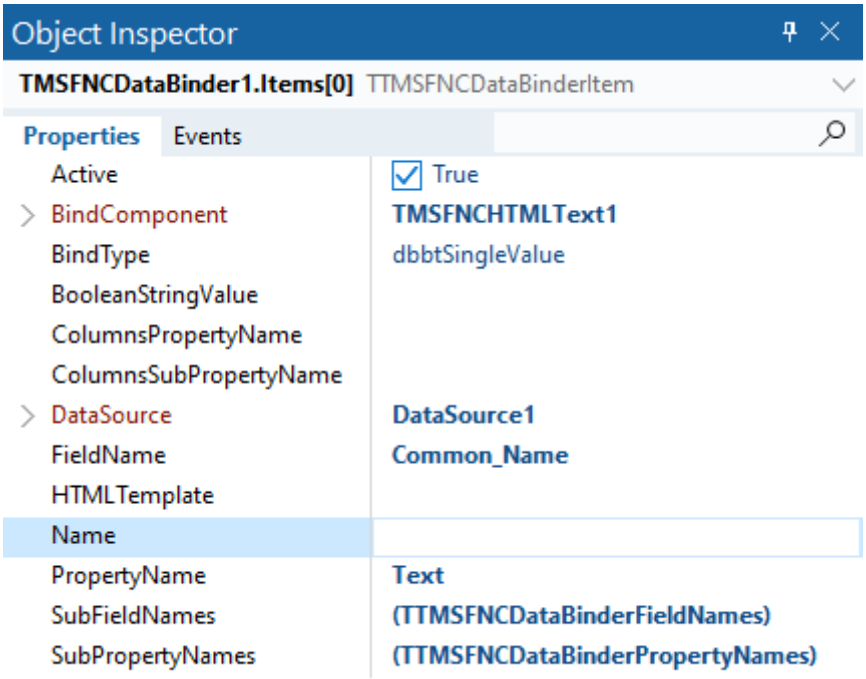
```
procedure TFormDataBinding.UpdateLinks;
var
  it: TTMSFNCDataBinderItem;
begin
  TMSFNCDataBinder1.BeginUpdate;
  it := TMSFNCDataBinder1.Items.Add;
  it.&Object := TMSFNCHTMLText1;
  it.BindType := dbbtSingleValue;
  it.DataSource := DataSource1;
  it.FieldName := 'Common_Name';
  it.PropertyName := 'Text';
  TMSFNCDataBinder1.EndUpdate;
  TMSFNCDataBinder1.Active := True;
end;
```

The below code actually sets up the same binding as the code above with a convenience method ConnectSingle.

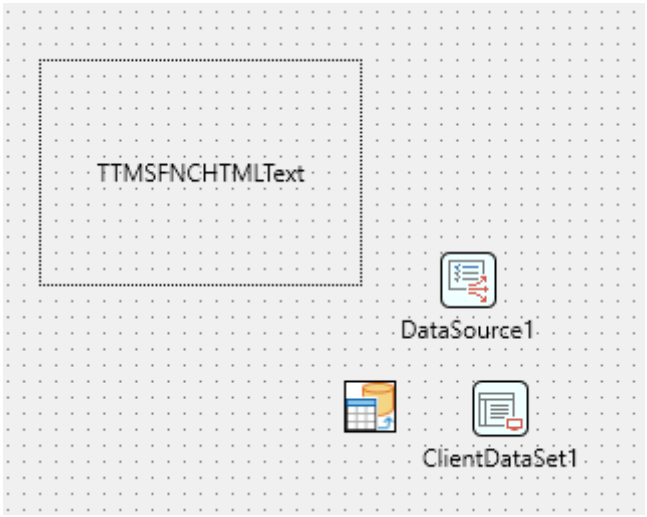
```
procedure TFormDataBinding.UpdateLinks;
begin
  TMSFNCDataBinder1.ConnectSingle(TMSFNCHTMLText1, DataSource1, 'Text', 'Common_Name');
  TMSFNCDataBinder1.Active := True;
end;
```

Designtime

The databinder component also registered designtime support retrieving components, fieldnames and properties and makes this easily selectable at designtime via the object inspector. Add an item to the Items collection inside the TTMSFNCDataBinder component, select the item in the object inspector and select the values necessary to bind a single value to the component as demonstrated below.



Result



Clown Triggerfish

Binding a list (TStringList)

Binding a list is similar to binding a single value but it shows all the records in the dataset.

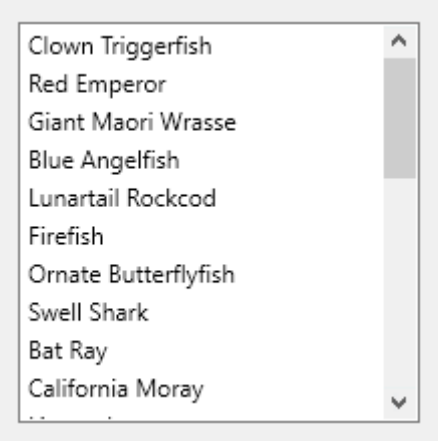
Programmatically

```
procedure TFormDataBinding.UpdateLinks;
var
  it: TTMSFNCDataBinderItem;
begin
  TMSFNCDataBinder1.BeginUpdate;
  it := TMSFNCDataBinder1.Items.Add;
  it.&Object := ListBox1;
  it.BindType := dbbtList;
  it.DataSource := DataSource1;
  it.FieldName := 'Common_Name';
  it.PropertyName := 'Items';
  TMSFNCDataBinder1.EndUpdate;
  TMSFNCDataBinder1.Active := True;
end;
```

Again as with the single value binding, the below code uses a convenience method to add an item and set all the properties in one go.

```
procedure TFormDataBinding.UpdateLinks;
begin
  TMSFNCDataBinder1.ConnectList(ListBox1, DataSource1, 'Items', 'Common_Name');
  TMSFNCDataBinder1.Active := True;
end;
```

Result



Binding a list (TCollection)

Binding a list based on TCollection is similar to binding a list based on TStringList. When binding a TStringList based component it is sufficient to set the property that represents the list (Items in case of a TListBox). Some components have a collection to represent the items and when using the above, it will add items to the collection but not visualize the content, as typically there is a Caption or Text property inside the TCollectionItem that is used to display the values inside the list.

To work with a TCollection instead of a TStringList in the databinder, you need to use the SubPropertyNames & the SubFieldNames collection and this immediately allows you to bind more than one sub-property. The PropertyName remains “Items”, but for each collection item that is automatically added to the “Items” collection, the SubPropertyNames & SubFieldNames collection specifies which properties of that item are bound to which fields. For each item inside the SubPropertyNames there is an equivalent in the SubFieldNames and vice versa. Below is a sample based on the TTMSFNCListBox component.

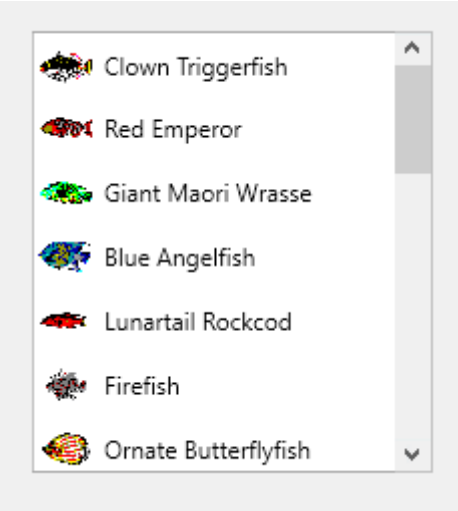
```
procedure TFormDataBinding.UpdateLinks;
var
  it: TTMSFNCDataBinderItem;
begin
  TMSFNCDataBinder1.BeginUpdate;
  it := TMSFNCDataBinder1.Items.Add;
  it.&Object := TMSFNCListBox1;
  it.BindType := dbbtList;
  it.DataSource := DataSource1;
  it.PropertyName := 'Items';
  it.SubPropertyNames.Add.Value := 'Text';
  it.SubFieldNames.Add.Value := 'Common_Name';
  TMSFNCDataBinder1.EndUpdate;
  TMSFNCDataBinder1.Active := True;
end;
```

or

```
procedure TFormDataBinding.UpdateLinks;
begin
  TMSFNCDataBinder1.ConnectList(TMSFNCListBox1, DataSource1, 'Items', ['Text'],
['Common_Name']);
  TMSFNCDataBinder1.Active := True;
end;
```

Below is a sample that demonstrates how to bind more then one property.

```
procedure TFormDataBinding.FormCreate(Sender: TObject);
begin
  TMSFNCListBox1.DefaultItem.BitmapWidth := 32;
  TMSFNCListBox1.DefaultItem.BitmapHeight := 32;
  TMSFNCListBox1.ItemsAppearance.HeightMode := lihmVariable;
  TMSFNCDataBinder1.ConnectList(TMSFNCListBox1, DataSource1, 'Items', ['Text', 'Bitmap'],
['Common_Name', 'Graphic']);
  TMSFNCDataBinder1.Active := True;
end;
```



Binding a column/list

Binding a column/list will retrieve all fields and all records and display them in a column/list structure. This means that the component you want to bind needs to have a separate column and a list collection.

```
procedure TFormDataBinding.FormCreate(Sender: TObject);
begin
  TMSFNCDataBinder1.ConnectColumnList(TMSFNCTreeView1, DataSource1, 'Nodes', 'Columns', 'Text',
'Values.Text');
  TMSFNCDataBinder1.Active := True;
end;
```

Species No	Category	Common_Nam	Species Name	Length (cm)	Length_In
90020	Triggerfishy	Clown Triggerfi	Ballistoides cor	50	19,6850393700
90030	Snapper	Red Emperor	Lutjanus sebae	60	23,6220472440
90050	Wrasse	Giant Maori Wr	Cheilinus undul	229	90,1574803149
90070	Angelfish	Blue Angelfish	Pomacanthus n	30	11,8110236220
90080	Cod	Lunartail Rockc	Variola louti	80	31,4960629921
90090	Scorpionfish	Firefish	Pterois volitans	38	14,9606299212
90100	Butterflyfish	Ornate Butterfl	Chaetodon Orr	19	7,48031496062
90110	Shark	Swell Shark	Cephaloscylliur	102	40,1574803149
90120	Ray	Bat Ray	Myliobatis calif	56	22,0472440944
90130	Eel	California Mora	Gymnothorax n	150	59,0551181102
90140	Cod	Lingcod	Ophiodon elon	150	59,0551181102
90150	Sculpin	Cabezon	Scorpaenichthy	99	38,9763779527
90160	Spadefish	Atlantic Spadef	Chaetodiperus	90	35,4330708661

Binding a grid

Binding a grid is only possible when the component you are targeting implements the ITMSFNCDataBinderGrid interface. You can either use the interface, or redefine the interface with the correct GUID.

```
ITMSFNCDataBinderBase = interface
  ['{778B63C9-34E3-4B65-A6B8-85E3EB1D17C3}']
  procedure DataBeginUpdate;
  procedure DataEndUpdate;
end;

ITMSFNCDataBinderGrid = interface(ITMSFNCDataBinderBase)
  ['{D23BDEAA-49B1-451A-9401-0D0D11A9957A}']
  procedure SetDataColumnCount(AValue: Integer);
  procedure SetDataRowCount(AValue: Integer);
  procedure ClearData;
  function GetDataRowCount: Integer;
  procedure SetDataValue(AColumn, ARow: Integer; AValue: string);
  procedure SetDataHeader(AColumn: Integer; AValue: string);
end;
```

Then after implementing the correct interfaces, you can use the following code to bind to a grid

```
procedure TFormDataBinding.FormCreate(Sender: TObject);
begin
  TMSFNCDataBinder1.ConnectGrid(TMSFNCGrid1, DataSource1);
  TMSFNCDataBinder1.Active := True;
end;
```

	Species No	Category	Common_N	Species Nar	Length (cm	Length_In	Notes	Gra
	90020	Triggerfishy	Clown Trigg	Ballistoides	50	19,68503937	(MEMO)	(GR
	90030	Snapper	Red Empero	Lutjanus seb	60	23,62204724	(MEMO)	(GR
	90050	Wrasse	Giant Maori	Cheilinus un	229	90,15748031	(MEMO)	(GR
	90070	Angelfish	Blue Angelfi	Pomacanthu	30	11,81102362	(MEMO)	(GR
	90080	Cod	Lunartail Ro	Variola louti	80	31,49606299	(MEMO)	(GR
	90090	Scorpionfish	Firefish	Pterois volit	38	14,96062992	(MEMO)	(GR
	90100	Butterflyfish	Ornate Butte	Chaetodon	19	7,480314960	(MEMO)	(GR
	90110	Shark	Swell Shark	Cephaloscyll	102	40,15748031	(MEMO)	(GR
	90120	Ray	Bat Ray	Myliobatis c	56	22,04724409	(MEMO)	(GR
	90130	Eel	California M	Gymnothora	150	59,05511811	(MEMO)	(GR
	90140	Cod	Lingcod	Ophiodon el	150	59,05511811	(MEMO)	(GR
	90150	Sculpin	Cabezon	Scorpaenich	99	38,97637795	(MEMO)	(GR
	90160	Spadefish	Atlantic Spa	Chaetodiper	90	35,43307086	(MEMO)	(GR

Binding a TList/TObjectList

Next to TStringList & TCollection there is also support for TList & TObjectList properties based on Integer & Strings. The sample below defines a TPersistent object with 2 published properties, bound to a data source.

```
  TListObject = class(TPersistent)
  private
    FL: TList<Integer>;
    FS: TList<string>;
  public
    constructor Create;
    destructor Destroy; override;
  published
    property L: TList<Integer> read FL;
    property S: TList<string> read FS;
   end;

procedure TFormDataBinding.Bind;
var
  o: TListObject;
begin
  o := TListObject.Create;
  try
    TMSFNCDataBinder1.Items.Clear;
    TMSFNCDataBinder1.ConnectList(o, DataSource1, 'L', [], ['Price']);
    TMSFNCDataBinder1.ConnectList(o, DataSource1, 'S', [], ['Brand']);

    o.Log;

    ClientDataSet1.First;
    ClientDataSet1.Edit;
    ClientDataSet1.Fields[1].AsString := 'Hello World!';
    ClientDataSet1.Post;

    o.Log;
  finally
    o.Free;
  end;
end;
```

Result

Debug Output:
{"$type":"TListObject","L":[699000,1268000,2096000,2000000,3881000,1659000,929000,9740500,690000,2010000,2035825,1175000,4779500,3950000,1685000,1259000,7410000,4537000,12445000,906500,1053000],"S":["Alfa Romeo","MERCEDES","TVR","Wiesmann","Honda","Lexus","Mazda","Rolls Royce","Audi","JAGUAR","MASERATI","LOTUS","FERRARI","BMW","PORSCHE","PEUGEOT","LAMBORGHINI","DE TOMASO","MERCEDES","MG","Chrysler"]} Process DataBindingTest.exe (1612)

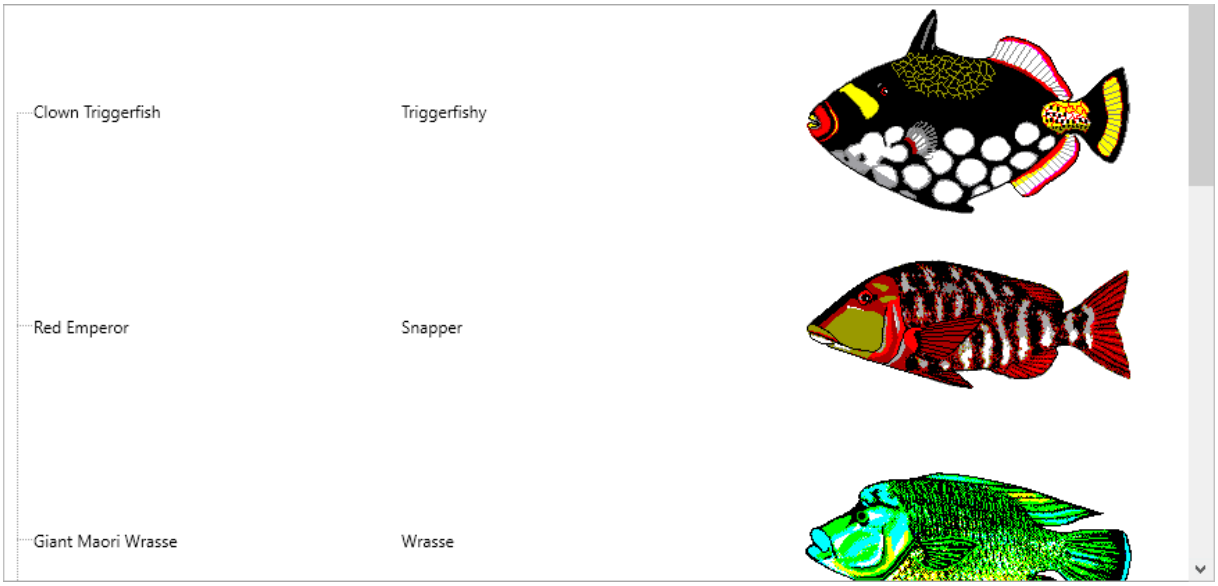
And the result after editing a value to “Hello World!”

Debug Output:
{"$type":"TListObject","L":[699000,1268000,2096000,2000000,3881000,1659000,929000,9740500,690000,2010000,2035825,1175000,4779500,3950000,1685000,1259000,7410000,4537000,12445000,906500,1053000],"S":["Hello World","MERCEDES","TVR","Wiesmann","Honda","Lexus","Mazda","Rolls Royce","Audi","JAGUAR","MASERATI","LOTUS","FERRARI","BMW","PORSCHE","PEUGEOT","LAMBORGHINI","DE TOMASO","MERCEDES","MG","Chrysler"]} Process DataBindingTest.exe (14732)

Sub collections

In the binding column/list chapter, you can see the sub property that is being targeted for the TTMSFNCTreeView “Nodes” collection is “Values.Text”. There is ofcourse no property named this way, but when activating the database adapter, the “Nodes” collection is filled with items. Each item represents a node (TTMSFNCTreeViewNode), and each node has a property called “Values”. This represents the value for each column. When adding a TTMSFNCTreeViewNodeValue inside the “Values” collection, you can set a “Text” property that will then be displayed in the treeview. To make this work we have added support in the database adapter to go a level deeper when necessary. When the sub property under the main property is also a TCollection, it will add a value and then access the property afterwards and set the value of the field accordingly.

TMSFNCDataBinder1.ConnectList(TMSFNCTreeView1, DataSource1, 'Nodes', ['Values.Text', 'Values.Text', 'Values.CollapsedIcon'], ['Common_Name', 'Category', 'Graphic']);



HTML Template

For the single value and list bindings, there is support to add a HTML template instead of binding to a specific field name. The HTML template supports multiple fields as long as they follow a specific kind of format. The HTML itself is based on the mini HTML reference (https://www.tmssoftware.com/site/minihtml.asp)

The format for adding fields is: <#FIELDNAME>.

So when applying this to the single value binding for example we can add an item using the

```
TMSFNCDataBinder1.ConnectSingleHTMLTemplate(TMSFNCHTMLText1, DataSource1, 'Text',
'<b>Name: <#COMMON_NAME></b><br/><#NOTES>');
```

or

```
procedure TFormDataBinding.UpdateLinks;
var
  it: TTMSFNCDataBinderItem;
begin
  TMSFNCDataBinder1.BeginUpdate;
  it := TMSFNCDataBinder1.Items.Add;
  it.&Object := TMSFNCHTMLText1;
  it.BindType := dbbtSingleValue;
  it.DataSource := DataSource1;
  it.PropertyName := 'Text';
  it.HTMLTemplate := '<b>Name: <#COMMON_NAME></b><br/><#NOTES>';
  TMSFNCDataBinder1.EndUpdate;
  TMSFNCDataBinder1.Active := True;
end;
```

This will result in:

Name: Clown Triggerfish
Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.

Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes rocked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering."

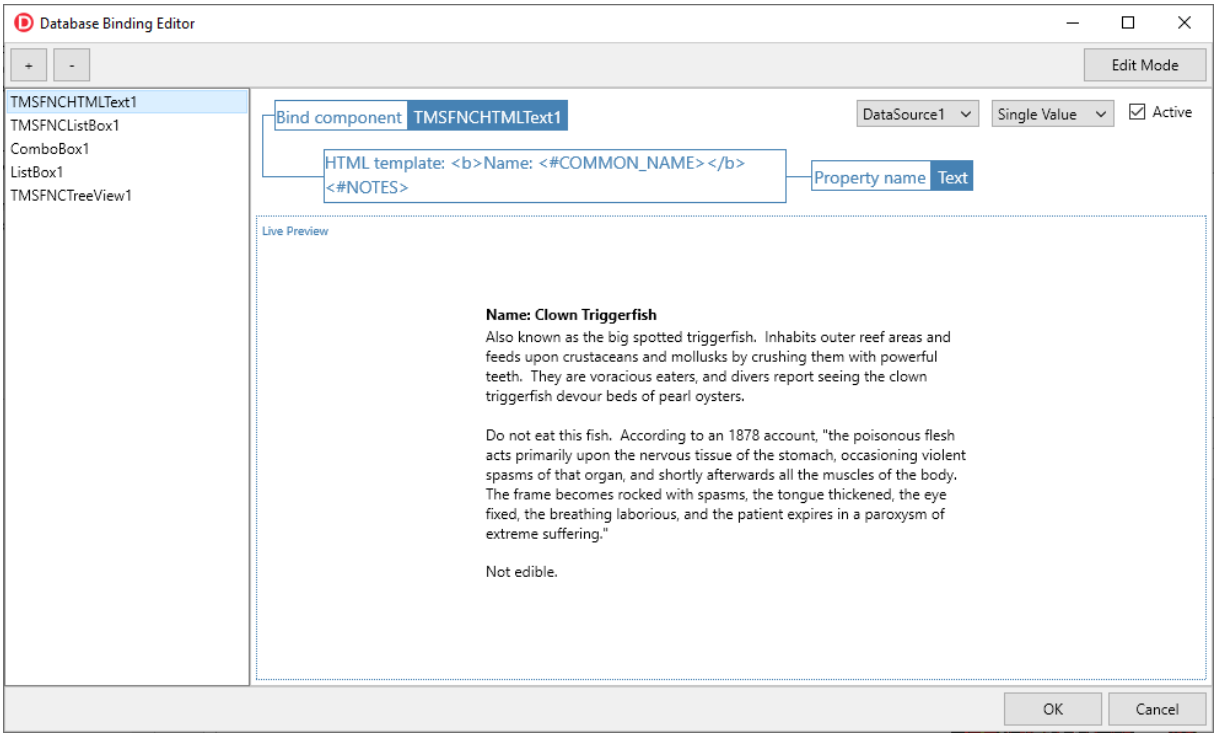
Not edible.

Editor

The databinder component installs an editor that is available at designtime and at runtime. To start it at runtime, call

TMSFNCDataBinder1.ShowEditor;

At designtime, you can right-click the editor and select “Edit…” to start the editor.



The editor allows you to edit, add and delete bindings. Click on Edit Mode to make changes to existing bindings. Click on the + sign at the top left to add a new binding and when in edit mode, select new components to start a new configuring a new binding. The comboboxes will list properties based on the bind component and the datasource. The datasource can be selected in the upper right corner next to the bind type.