you will need basic knowledge of the DataGrid

How to use the DLL:

1. Step

Add the DLL as a reference to your project

2. Step

Include the namespace:

```
using DynTemplateColumns.NCFDataGrid;
```

3. Step

from any function inside the Page Init Process e.g. in OnInit:

```
override protected void OnInit(EventArgs e)
{
	//
	// CODEGEN: This call is required by the ASP.NET Web Form Designer.
	//
	InitializeComponent();

		...

	SuperGrid_Init();

	if(!IsPostBack)
		SuperGrid_Bind();

		...

	base.OnInit(e);
}
```

most convincing in a separate function or class, create the columns:

```
private void SuperGrid_Init()
{

	SuperGrid.DataKeyField="id";

			//normal bound column
	BoundColumn	bc_ID= CFCustomColumn.CreateBoundColumn("id","id",true);
	SuperGrid.Columns.Add(bc_ID);

			//normal bound column, but shorter
		SuperGrid.Columns.Add(CFCustomColumn.CreateBoundColumn("author","name",false));

	...


		//here it happens
	TemplateColumn myCol=CFCustomColumn.CreateTemplateColumn(
		(new Label()).GetType(),  /*yes i know, i'm lazy!*/
		new CFEventhandlers.CustomColumnItemDelegate(SuperGrid_CreateLabel),
		new CFEventhandlers.CustomColumnItemDelegate(SuperGrid_BindLabel),
		(new DropDownList()).GetType(),
		new CFEventhandlers.CustomColumnItemDelegate(SuperGrid_CreateDropDown),
		new CFEventhandlers.CustomColumnItemDelegate(SuperGrid_BindDropDown));

	myCol.HeaderText="type";
	SuperGrid.Columns.Add(myCol);

	...

}
```

The CreateTemplateColumn defined as:

```
public static TemplateColumn CreateTemplateColumn(
			Type ItemControlType,
			CFEventhandlers.CustomColumnItemDelegate ItemCreateEvent,
			CFEventhandlers.CustomColumnItemDelegate ItemBindEvent,
			Type EditItemControlType,
			CFEventhandlers.CustomColumnItemDelegate EditItemCreateEvent,
			CFEventhandlers.CustomColumnItemDelegate EditItemBindEvent)
```

Every Column in a DataGrid gets a (WebControl)Type for the normal view and for the edit mode. If you don’t need edit mode, you can set its type and Eventhandlers to null.

4. Step

The Eventhandlers:

They are defined as:

```
public delegate void CustomColumnItemDelegate(object sender, CustomColumnItemEventArgs e);
```

object sender - will be the Control

CustomColumnItemEventArgs args - is a special class, which derived from the standard EventArgs.

There are 4 Eventhandlers, you can use:

a) When DataRow is in normal view:

1. ItemCreateEvent – gets called for each row, after the control is created, but not added to the DataGrid

2. ItemBindEvent - gets called for each row, when DataGrid is in binding process

b) When DataRow is in edit mode:

1. EditItemCreateEvent - gets called for the selected row to edit, after the control is created, but not added to the DataGrid

2. EditItemBindEvent - gets called for the selected row to edit, when DataGrid is in binding process

Set up the Eventhandlers, sample:

Note the following functions contain Pseudo-Code or function calls not defined in this document, they are marked with this color.

```
private void SuperGrid_CreateLabel(object sender, CustomColumnItemEventArgs args)
{
	//you can change the control properties here
}


private void SuperGrid_BindLabel(object sender, CustomColumnItemEventArgs args)
{
	//get index of sender and pageindex of DataGrid, and use them to receive the
	//appropriate row and data in your DataSet/DataTable, use caching if possible
	string data=GetDataFromDataBase((Label)sender, args.e.ItemIndex);

	((Label)sender).Text = data;

	//you can change the control properties here

}


private void SuperGrid_CreateDropDown(object sender, CustomColumnItemEventArgs args)
{
	//you can change the control properties here
}
```

```
private void SuperGrid_BindDropDown(object sender, CustomColumnItemEventArgs args)
{
	//if this fails something is going really bad for you
	DropDownList x= ((DropDownList)sender);

	//get index of sender and pageindex of DataGrid, and use them to receive the
	//appropriate rowindex
	int masterrowindex = GetItemIndex(x);
	if(masterrowindex < (int)0)
		return;

	//Tables
	DataTable detailTable = GetDropDownTableFromDB();
	DataTable masterTable = GetGridTableFromDB();

	//get foreign-id value in master datatable
	string strFID=GetTableValue("fid",masterrowindex,masterTable).ToString();

	//get column index in detail table
	int indexID=detailTable.Columns.IndexOf("id");
	int indexName=detailTable.Columns.IndexOf("name");

	//loop detailTable and populate the DropDown x
	for(int rowindex=0; rowindex < detailTable.Rows.Count; rowindex++)
	{

		ListItem li = new ListItem();
		li.Value=detailTable.Rows[rowindex].ItemArray.GetValue(indexID).ToString();
		li.Text=detailTable.Rows[rowindex].ItemArray.GetValue(indexName).ToString();
		x.Items.Add(li);

		//important, select the correct item !
		if(strFID == li.Value)
			x.SelectedIndex=rowindex;
	}

	//please note, that this function is only a demonstration on how to do, in real world
	//you should do data retrieval once and then use caching/update features on some parts,
	//instead of looping again and again
}
```