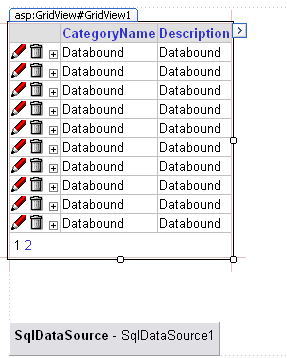
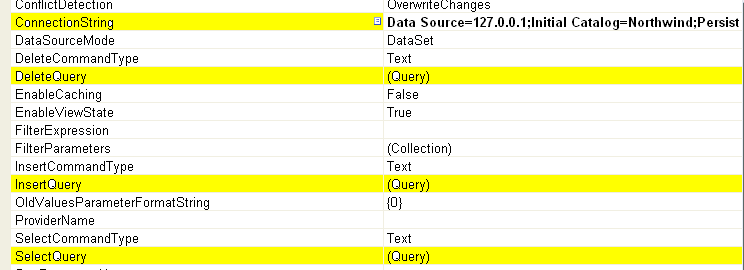
Se desea crear un maestro detalle como el siguiente



**PAGINA MASTER.ASPX**



**Control: SqlDataSource**



**ConnectionString:**

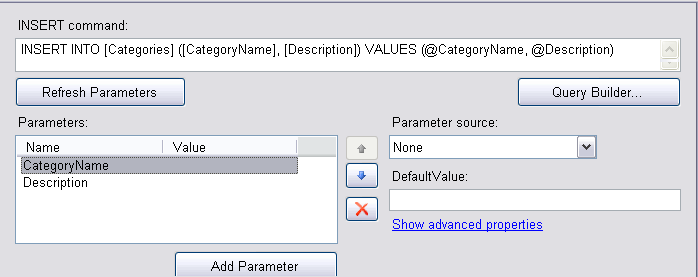
Data Source=127.0.0.1;Initial Catalog=Northwind;Persist Security Info=True;User ID=sa;Password=jill

**SelectQuery**

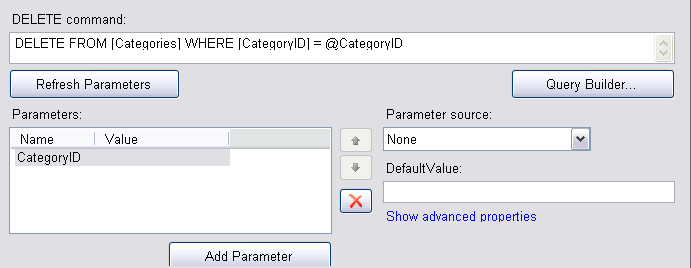
SELECT '0' as [CategoryID], '' as [CategoryName], '' as [Description] UNION SELECT [CategoryID], [CategoryName], **convert(varchar(100),[Description]) as [Description]** FROM [Categories] ORDER BY [CategoryName]

**InsertQuery**

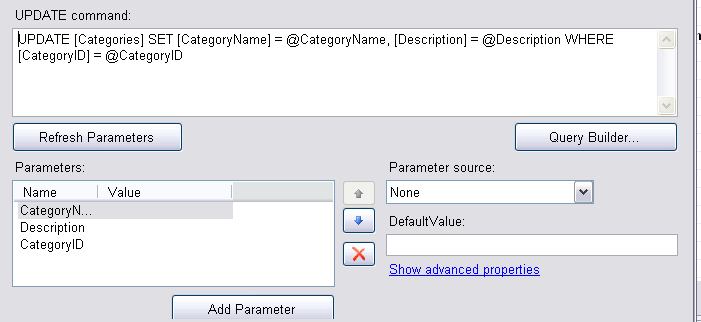
SELECT '0' as [CategoryID], '' as [CategoryName], '' as [Description] UNION SELECT [CategoryID], [CategoryName], convert(varchar(100),[Description]) as [Description] FROM [Categories] ORDER BY [CategoryName]



**DeleteQuey**

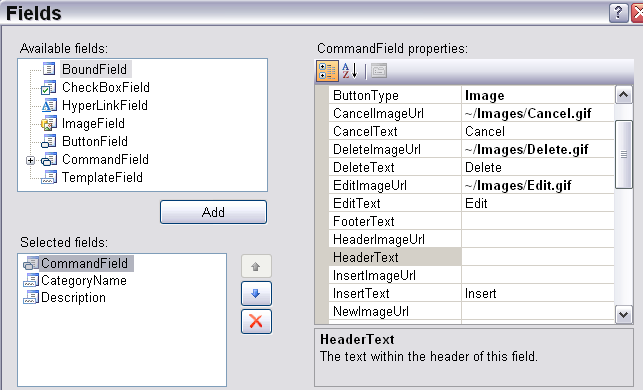


**UpdateQuey**



**Control: DataView**

**Property: collections**



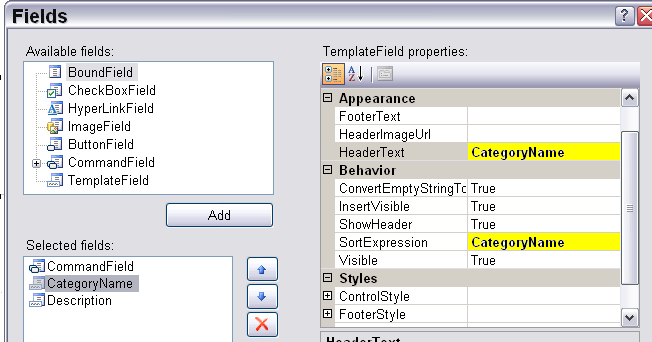
**Tiene 3 opciones, 1 comando y 2 campos**

**El command field tiene**

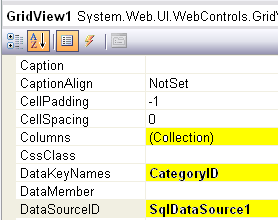
**Cancel Text y CancelImageURL**

* **EditText y EditImageURL**
* **InsertText y InsertImageURL**
* **DeleteText y DeleteImageURL**
* **SelectedText y SelectedImageURL (para dirigirse al detalle, el signo (+)**

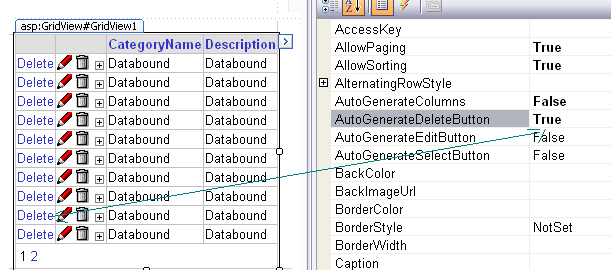
Si es campo de tipo BoundField como CategoryName



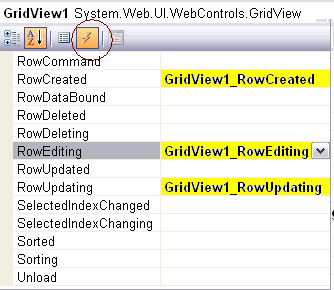
Las otras opciones importantes para dar un Bind son: **DataKeyNames y DataSourceID**



Podria generarse automaticamente los botones de insertar, borrar, actualizar



Se definen 3 eventos



**RowCreated (para generar el detalle)**

protected void GridView1\_RowCreated(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.Header) **AddGlyph**(GridView1, e.Row);

if (e.Row.RowType == DataControlRowType.DataRow)

{

string KeyValue = GridView1.DataKeys[e.Row.RowIndex].Value.ToString();

if (KeyValue == "0" && EditIndex == -1) // we are not editing

e.Row.Attributes.Add("isadd", "1");

//create Url to be executed when the "+" is clicked

string RowID = Convert.ToString(System.Web.UI.DataBinder.Eval(e.Row.DataItem, "CategoryID"));

**string Url = "Details.aspx?ID=" + RowID;**

**e.Row.Attributes.Add("href", Url); // link to details**

// used by the detail table expander/contracter

e.Row.Attributes.Add("open", "0");

// used to prevent excessive callbacks to server

e.Row.Attributes.Add("hascontent", "0"); }

}

void AddGlyph(GridView grid, GridViewRow item)

{

if (grid.AllowSorting == false) return;

Label glyph = new Label();

glyph.EnableTheming = false;

glyph.Font.Name = "webdings";

glyph.Font.Size = FontUnit.XSmall;

glyph.Text = (grid.SortDirection == SortDirection.Ascending ? "5" : " 6");

// Find the column you sorted by

for (int i = 0; i < grid.Columns.Count; i++)

{

string colExpr = grid.Columns[i].SortExpression;

if (colExpr != "" && colExpr == grid.SortExpression)

{

item.Cells[i].Controls.Add(glyph);

}

}

}

**PARA INICIAR EDITAR**

public int EditIndex = -1;

protected void GridView1\_RowEditing(object sender, GridViewEditEventArgs e)

{

EditIndex = e.NewEditIndex;

}

**PARA ACTUALIZAR EDICION**

protected void GridView1\_RowUpdating(object sender, GridViewUpdateEventArgs e)

{

string KeyValue = GridView1.DataKeys[e.RowIndex].Value.ToString();

if (KeyValue != "0")return; // key value of 0 indicates the insert row

System.Web.UI.WebControls.SqlDataSource ds = (System.Web.UI.WebControls.SqlDataSource)this.FindControl(this.GridView1.DataSourceID);

System.Data.SqlClient.SqlConnection conn = new System.Data.SqlClient.SqlConnection(ds.ConnectionString);

conn.Open();

string s = ds.InsertCommand;

System.Data.SqlClient.SqlCommand c = new System.Data.SqlClient.SqlCommand(s, conn);

System.Data.SqlClient.SqlParameter p;

foreach (System.Collections.DictionaryEntry x in e.NewValues)

{

p = new System.Data.SqlClient.SqlParameter("@" + x.Key, x.Value);

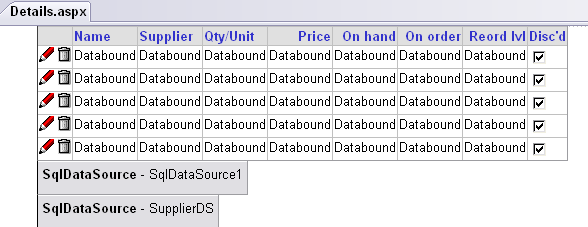
c.Parameters.Add(p);

}

c.ExecuteNonQuery();

}

**PAGINA DETAIL.ASPX**



**Control: SqlDataSource1 y SupplierDS**

**El SqlDataSource1 es para el grid, el otro es para actualizar el Supplier al editar**

**SelectQuery**

**SELECT '0' as [ProductID], '' as [ProductName], '' as [CompanyName], '0' as SupplierID, '0' as [QuantityPerUnit], '0.00' as [UnitPrice], '0' as [UnitsInStock], '0' as [UnitsOnOrder], '0' as [ReorderLevel], '0' as [Discontinued]**

**UNION**

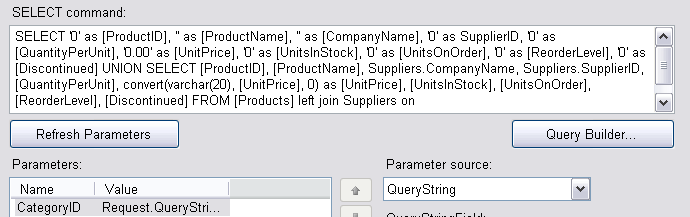
**SELECT [ProductID], [ProductName], Suppliers.CompanyName, Suppliers.SupplierID, [QuantityPerUnit], convert(varchar(20), [UnitPrice], 0) as [UnitPrice], [UnitsInStock], [UnitsOnOrder], [ReorderLevel], [Discontinued]**

**FROM [Products] left join Suppliers on Suppliers.Supplierid=Products.SupplierID**

**WHERE ([CategoryID] = @CategoryID) ORDER BY [ProductName]**

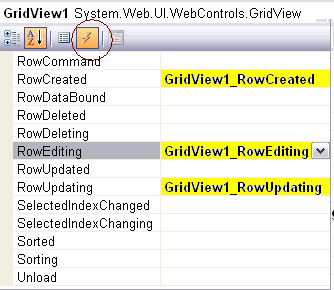
**El convert del dinero es necesario para evitar problemas que no es formato dinero**

**Se debe dar un Union por si no tiene data aparezca cero**



Puede agregarse InsertQuery, DeleteQuery, UpdateQuery

Se definen 3 eventos



**RowCreated (para generar el detalle)**

protected void GridView1\_RowCreated(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.Header) AddGlyph(GridView1, e.Row);

if (e.Row.RowType == DataControlRowType.DataRow)

{

string KeyValue = GridView1.DataKeys[e.Row.RowIndex].Value.ToString();

// we are not editing

if (KeyValue == "0" && EditIndex == -1) e.Row.Attributes.Add("isadd", "1");

}

}

void AddGlyph(GridView grid, GridViewRow item)

{

if (grid.AllowSorting == false) return;

Label glyph = new Label();

glyph.EnableTheming = false;

glyph.Font.Name = "webdings";

glyph.Font.Size = FontUnit.XSmall;

glyph.Text = (grid.SortDirection == SortDirection.Ascending ? "5" : " 6");

// Find the column you sorted by

for (int i = 0; i < grid.Columns.Count; i++)

{

string colExpr = grid.Columns[i].SortExpression;

if (colExpr != "" && colExpr == grid.SortExpression)

{

item.Cells[i].Controls.Add(glyph);

}

}

}

**PARA INICIAR EDITAR**

public int EditIndex = -1;

protected void GridView1\_RowEditing(object sender, GridViewEditEventArgs e)

{

EditIndex = e.NewEditIndex;

}

**PARA ACTUALIZAR EDICION**

protected void GridView1\_RowUpdating(object sender, GridViewUpdateEventArgs e)

{

string KeyValue = GridView1.DataKeys[e.RowIndex].Value.ToString();

if (KeyValue != "0") return; // key value of 0 indicates the insert row

System.Web.UI.WebControls.SqlDataSource ds = (System.Web.UI.WebControls.SqlDataSource)this.FindControl(this.GridView1.DataSourceID);

System.Data.SqlClient.SqlConnection conn = new System.Data.SqlClient.SqlConnection(ds.ConnectionString);

conn.Open();

string s = ds.InsertCommand;

System.Data.SqlClient.SqlCommand c = new System.Data.SqlClient.SqlCommand(s, conn);

System.Data.SqlClient.SqlParameter p;

foreach (System.Collections.DictionaryEntry x in e.NewValues)

{

p = new System.Data.SqlClient.SqlParameter("@" + x.Key, x.Value);

c.Parameters.Add(p);

}

**p = new System.Data.SqlClient.SqlParameter("@CategoryID", Request.QueryString["ID"]);**

**c.Parameters.Add(p);**

**c.ExecuteNonQuery();**

}

A diferencia del Master, se debe actualizar y se toma como referencia el CategoryID del QueryStringPara agregar el efecto que cuando se escoge los registros del detalle no cargue todas las veces (sea mas rapido), cuando colapsa solo los esconde.

**Master.aspx**

Inserte este codigo en la seccion de <head>

<script src="GridViewHelper.js" type="text/javascript"></script>

<link href="GridViewHelper.css" rel="STYLESHEET" type="text/css" />

<link href="AppStyle.css" rel="STYLESHEET" type="text/css" />

Inserte el codigo en la seccion de </form>

<script type="text/javascript">

GridViewHelper.Init(document.all.GridView1, 0, 0);

var ToolTips = new Array("", "??", "??", "??");

GridViewHelper.AddToolTips(document.all.GridView1, ToolTips);

</script>

* Since this is a master page and not a detail page (the code detects that it is not running in an IFRAME), the Init function resizes the width of the page to the required width so that no columns wrap. If you don't want this functionality, don't call Init in the master page. The other parameters are required but ignored.
* The GridView control does not support tooltips on the column headers, so GridViewHelper has a method AddToolTips which takes a reference to the grid and an array of tooltips that it applies to the column headers. Change "??" to whatever you want.

**Details.aspx**

Notice that the <body> has been modified. This is to force the details to "snug" up against the master record and offset to the right by 40 pixels to allow the details table to appear indented. Also, scrolling has been turned off. You can safely turn scrolling off because the master page has been widened. Again, this is to "snug" up the detail records:

<body **leftmargin="40" topmargin="0" scroll="no"**>

Insert this code in the <head> section of the file:

<script src="GridViewHelper.js" type="text/javascript"></script>

<link href="GridViewHelper.css" rel="STYLESHEET" type="text/css" />

<link href="AppStyle.css" rel="STYLESHEET" type="text/css" />

Notice that the <form> has been modified.

<form id="form1" runat="server" **style="overflow: hidden;"**>

Notice that the <div> has been modified.

<div **style="overflow: hidden;"**>

And now the **magic**. Insert this code following the **</form>** element:

<script type="text/javascript">

GridViewHelper.Init(document.all.GridView1,

ExtraWidth=100, ExtraHeight=0);

var ToolTips = new Array("", "Product name",

"Supplier", "Quantity per unit",

"Unit price", "Units on order",

"Reorder level", "Discontinued");

GridViewHelper.AddToolTips(document.all.GridView1, ToolTips);

</script>

This causes the IFRAME, where the details are placed in, to be only as big as necessary. The ultimate "snugging". Notice the ExtraWidth=100, ExtraHeight=0 parameters above. When scroll in the <body> is set to no, you have nothing to do as these parameter values should suffice. If you set scroll="yes", then scroll bars may appear and you have to make room for them. This is where you experiment with changing the ExtraWidth/ExtraHeight parameters to larger numbers so that the detail records look correct (as few scroll bars as necessary). You never need a vertical scroll bar because the code sizes to the correct height. If you change ExtraHeight=25, you should have enough room for a horizontal scroll bar without the need for a vertical scroll bar.