

# Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.aspnet.buildingcontrols/2006-12/ms>

---

- *From:* "schmurgon" <[schmurgon@xxxxxxxxxx](mailto:schmurgon@xxxxxxxxxx)>
  - *Date:* 15 Dec 2006 22:03:00 -0800
- 

Hi,

This has been driving me nuts for 2 days now. I have a custom control derived from CompositeDataBoundControl that contains a GridView control and a few other controls. I am wanting to expose a "Coumns" property on my control that uses the same designer that the GridView uses for it's Columns property. I have managed to get this far, however after adding the desired columns in design-time no changes are persisted. What am I doing wrong? From what I have read, nested controls look after their own ControlState e.t.c. It's as if I'm missing an event hook that should fire when the "OK" button in the DataControlFieldTypeEditor is pressed.

```
namespace Schmurgon.WebControls
{
    public enum ContainerType
    {
        ActivityLogSummaryGridView = 1,
        ActivityDescriptionLabel = 2,
        ActivityDetailLabel = 3,
        CreateNoteButton = 4
    }

    /// <summary>
    /// Container control for individual components of the
    /// ActivityLog control
    /// </summary>
    public class ActivityLogContainer : TableCell, INamingContainer
    {
        private ContainerType _containerType;

        /// <summary>
        /// Creates a new ActivityLogContainer of the specified type
        /// </summary>
        /// <param name="containerType"></param>
        public ActivityLogContainer( ContainerType containerType )
        {
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
_containerType = containerType;
}

/// <summary>
/// The container type
/// </summary>
public ContainerType ContainerType
{
    get { return _containerType; }
}

[DefaultProperty("Text")]
[ToolboxData("<{0}:ActivityLog runat=server></{0}:ActivityLog>")]
public class ActivityLog : CompositeDataBoundControl,
IPostBackEventHandler
{
    #region Fields

    private Label _activityDetailLabel;
    private Label _activityDescriptionLabel;
    private GridView _activityLogSummaryGridView;
    private Button _createNoteButton;
    private ActivityLogContainer _activityDetailLabelContainer;
    private ActivityLogContainer
    _activityDescriptionLabelContainer;
    private ActivityLogContainer
    _activityLogSummaryGridViewContainer;
    private ActivityLogContainer _createNoteButtonContainer;

    private static readonly object CommandEventKey = new object();

    #endregion

    #region Properties

    [Bindable(true)]
    [Category("Data")]
    [DefaultValue("")]
    [Localizable(true)]
    [NotifyParentProperty(true)]
    /// <summary>
    /// Gets or sets the ID of the DataSource control
    /// </summary>
    public override string DataSourceID
    {
        get
        {
            EnsureChildControls();
            return _activityLogSummaryGridView.DataSourceID;
        }
    }
}

```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
}  
set  
{  
EnsureChildControls();  
_activityLogSummaryGridView.DataSourceID = value;  
}  
}
```

```
[Bindable(true)]  
[Category("Appearance")]  
[DefaultValue("")]  
[Localizable(true)]  
public virtual string ActivityDescriptionText  
{  
get  
{  
EnsureChildControls();  
_activityDescriptionLabel.Text =  
(String)ViewState["ActivityDescriptionLabelText"];  
return ((_activityDescriptionLabel.Text == null) ?  
String.Empty : _activityDescriptionLabel.Text );  
}  
set  
{  
EnsureChildControls();  
ViewState["ActivityDescriptionLabelText"] = value;  
_activityDescriptionLabel.Text = value;  
}  
}
```

```
[Bindable(true)]  
[Category("Appearance")]  
[DefaultValue("")]  
[Localizable(true)]  
public virtual string ButtonText  
{  
get  
{  
EnsureChildControls();  
_createNoteButton.Text =  
(String)ViewState["CreateNoteButtonText"];  
return ((_createNoteButton.Text == null) ? String.Empty  
: _createNoteButton.Text);  
}  
set  
{  
EnsureChildControls();  
ViewState["CreateNoteButtonText"] = value;  
_createNoteButton.Text = value;  
}  
}
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
public string CommandName
{
    get { return _createNoteButton.CommandName; }
}

public string CommandArgument
{
    get { return _createNoteButton.CommandArgument; }
}

[PersistenceMode(PersistenceMode.InnerProperty)]
[Editor(typeof(
System.Web.UI.Design.WebControls.DataControlFieldTypeEditor), typeof(
UITypeEditor ))]
[MergableProperty(false)]
[DefaultValue( (string)null)]
[Category("Default")]
[DesignerSerializationVisibility(
DesignerSerializationVisibility.Content)]
public virtual DataControlFieldCollection Columns
{
    get
    {
        EnsureChildControls();
        return _activityLogSummaryGridView.Columns;
    }
}

[Editor(typeof(
System.Web.UI.Design.WebControls.DataControlFieldTypeEditor ), typeof(
UITypeEditor )), TypeConverter( typeof( StringArrayConverter ))]
[Category("Data")]
public string[] DataKeyNames
{
    get
    {
        return _activityLogSummaryGridView.DataKeyNames;
    }
}

#endregion

#region Events

public event CommandEventHandler Command
{
    add { Events.AddHandler( CommandEventKey, value );}
    remove { Events.RemoveHandler( CommandEventKey, value );}
}


```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
}  
  
#endregion  
  
protected override void OnInit( EventArgs e )  
{  
    EnsureChildControls();  
    base.OnInit( e );  
    InitializeComponent();  
}  
  
private void InitializeComponent()  
{  
    _activityLogSummaryGridView.RowCommand += new  
    GridViewCommandEventHandler(  
    ActivityLogSummaryGridView_RowCommand );  
  
    _activityLogSummaryGridView.RowDataBound += new  
    GridViewRowEventHandler(  
    ActivityLogSummaryGridView_RowDataBound );  
  
    _activityLogSummaryGridView.RowCreated += new  
    GridViewRowEventHandler(  
    ActivityLogSummaryGridView_RowCreated );  
  
    _activityLogSummaryGridView.Columns.FieldsChanged +=  
    new EventHandler(  
    ActivityLogSummaryGridView_FieldsChanged );  
}  
  
private void ActivityLogSummaryGridView_RowCreated( object  
sender, GridViewRowEventArgs e )  
{  
    // do Something  
}  
  
void ActivityLogSummaryGridView_RowCommand( object sender,  
GridViewCommandEventArgs e )  
{  
    // Do Something  
}  
  
private void ActivityLogSummaryGridView_RowDataBound( object  
sender, GridViewRowEventArgs e )  
{  
    if ( e.Row.RowType == DataControlRowType.DataRow )  
    {  
        // Do Something  
    }  
}
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
private void ActivityLogSummaryGridView_FieldsChanged( object
sender, EventArgs e )
{
}
}
```

```
protected virtual void OnCommand( CommandEventArgs e )
{
CommandEventHandler handler = Events[ CommandEventKey ] as
CommandEventHandler;
```

```
if ( handler != null )
{
handler( this, e );
}
}
```

```
RaiseBubbleEvent( this, e );
}
```

```
/// <summary>
/// Create a new ActivityLogContainer
/// </summary>
/// <param name="containerType"></param>
/// <returns></returns>
protected virtual ActivityLogContainer CreateContainer(
ContainerType containerType )
{
return new ActivityLogContainer( containerType );
}
}
```

```
/// <summary>
///
/// </summary>
/// <param name="activityLogContainer"></param>
protected virtual void CreateContainerChildControls(
ActivityLogContainer activityLogContainer )
{
switch ( activityLogContainer.ContainerType )
{
case ContainerType.ActivityDetailLabel:
_activityDetailLabel = new Label();
_activityDetailLabel.Width = Unit.Percentage(100);
_activityDetailLabel.ID = "_activityDetailLabel";
_activityDetailLabel.Text = "[Activity Detail]";
activityLogContainer.Controls.Add(
_activityDetailLabel );
break;
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
case ContainerType.ActivityDescriptionLabel:
    _activityDescriptionLabel = new Label();
    _activityDescriptionLabel.Width =
    Unit.Percentage(100);
    _activityDescriptionLabel.ID =
    "_activityDescriptionLabel";
    _activityDescriptionLabel.Text = "[Description]";

    activityLogContainer.Controls.Add(_activityDescriptionLabel);
    break;

case ContainerType.ActivityLogSummaryGridView:
    _activityLogSummaryGridView = new GridView();
    _activityLogSummaryGridView.Visible = true;
    _activityLogSummaryGridView.EmptyDataText = "No
    Activities";
    _activityLogSummaryGridView.Width =
    Unit.Percentage(100);
    _activityLogSummaryGridView.ID =
    "_activityLogSummaryGridView";
    // _activityLogSummaryGridView.DataSourceID =
    DataSourceID;
    _activityLogSummaryGridView.AutoGenerateColumns =
    false;
    _activityLogSummaryGridView.DataKeyNames = new
    string[] { "ActivityId" };

    // InitializeGridViewColumns();

    activityLogContainer.Controls.Add(
    _activityLogSummaryGridView );
    break;

case ContainerType.CreateNoteButton:
    _createNoteButton = new Button();
    _createNoteButton.CommandName = "CreateNote";
    _createNoteButton.Text = ButtonText;
    _createNoteButton.ID = "_createNoteButton";
    activityLogContainer.Controls.Add(
    _createNoteButton );
    break;
}
}

private void InitializeGridViewColumns()
{
    BoundField activityLogId = new BoundField();
    BoundField description = new BoundField();
    BoundField detail = new BoundField();
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
activityLogId.HeaderText = "ID";
activityLogId.DataField = "ActivityId";
activityLogId.InsertVisible = false;
activityLogId.SortExpression = "ActivityId";

description.HeaderText = "Description";
description.DataField = "Description";
description.SortExpression = "Description";

detail.HeaderText = "Detail";
detail.DataField = "Detail";
detail.SortExpression = "Detail";

_activityLogSummaryGridView.Columns.Add(activityLogId);
_activityLogSummaryGridView.Columns.Add(description);
_activityLogSummaryGridView.Columns.Add(detail);
}

/// <summary>
///
/// </summary>
protected virtual void ApplyContainerStyles()
{
    ApplyContainerStyle( _activityDetailLabelContainer );
    ApplyContainerStyle( _activityDescriptionLabelContainer );
    ApplyContainerStyle( _activityLogSummaryGridViewContainer
);
    ApplyContainerStyle( _createNoteButtonContainer );
}

/// <summary>
///
/// </summary>
/// <param name="container"></param>
private void ApplyContainerStyle( ActivityLogContainer
container )
{
    container.VerticalAlign = VerticalAlign.Middle;
    container.Wrap = true;
    container.BackColor = Color.FromName( "White" );
    container.ForeColor = Color.FromName( "Black" );

    if ( container.ContainerType ==
ContainerType.CreateNoteButton )
    {
        container.HorizontalAlign = HorizontalAlign.Right;
    }
    else
    {
        container.HorizontalAlign = HorizontalAlign.Left;
    }
}
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
}  
  
protected virtual void AddContainer( ActivityLogContainer  
container )  
{  
Controls.Add( container );  
}  
  
protected virtual void RenderContainer( ActivityLogContainer  
container,  
HtmlTextWriter writer )  
{  
container.RenderControl( writer );  
}  
  
protected override void CreateChildControls()  
{  
Controls.Clear();  
_activityDetailLabelContainer = CreateContainer(  
ContainerType.ActivityDetailLabel );  
CreateContainerChildControls( _activityDetailLabelContainer  
);  
AddContainer( _activityDetailLabelContainer );  
  
_activityDescriptionLabelContainer = CreateContainer(  
ContainerType.ActivityDescriptionLabel );  
CreateContainerChildControls(  
_activityDescriptionLabelContainer );  
AddContainer( _activityDescriptionLabelContainer );  
  
_activityLogSummaryGridViewContainer = CreateContainer(  
ContainerType.ActivityLogSummaryGridView );  
CreateContainerChildControls(  
_activityLogSummaryGridViewContainer );  
AddContainer( _activityLogSummaryGridViewContainer );  
  
_createNoteButtonContainer = CreateContainer(  
ContainerType.CreateNoteButton );  
CreateContainerChildControls( _createNoteButtonContainer );  
AddContainer( _createNoteButtonContainer );  
  
ChildControlsCreated = true;  
}  
  
/// <summary>  
///  
/// </summary>  
protected override HtmlTextWriterTag TagKey  
{  
get { return HtmlTextWriterTag.Table; }  
}
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
/// <summary>
///
/// </summary>
/// <returns></returns>
protected override Style CreateControlStyle()
{
return new TableStyle( ViewState );
}

/// <summary>
///
/// </summary>
/// <param name="writer"></param>
protected override void RenderContents( HtmlTextWriter writer )
{
ApplyContainerStyles();
writer.RenderBeginTag( HtmlTextWriterTag.Tr );
RenderContainer( _activityLogSummaryGridViewContainer,
writer );
writer.RenderEndTag();

writer.RenderBeginTag( HtmlTextWriterTag.Tr );
RenderContainer( _activityDescriptionLabelContainer, writer
);
writer.RenderEndTag();

writer.RenderBeginTag( HtmlTextWriterTag.Tr );
RenderContainer( _activityDetailLabelContainer, writer );
writer.RenderEndTag();

writer.RenderBeginTag( HtmlTextWriterTag.Tr );
RenderContainer( _createNoteButtonContainer, writer );
writer.RenderEndTag();
}

/// <summary>
///
/// </summary>
/// <param name="source"></param>
/// <param name="args"></param>
/// <returns></returns>
protected override bool OnBubbleEvent(object source, EventArgs
args)
{
bool handled = false;

//TODO: Implement this if we want to raise our own events,
and not expose the button
//command event
CommandEventArgs commandEventArgs = args as
```

## Persisting Columns property in Designer for a GridView inside a CompositeDataBoundControl

```
CommandEventArgs;
if ( commandEventArgs != null &&
commandEventArgs.CommandName == "CreateNote" )
{
// Create new CreateNoteEventArgs class e.t.c.
OnCommand( commandEventArgs );
handled = true;
}

return handled;
}

void IPostBackEventHandler.RaisePostBackEvent( string
eventArgument )
{
RaisePostBackEvent( eventArgument );
}

protected virtual void RaisePostBackEvent( string eventArgs )
{
CommandEventArgs e = new CommandEventArgs( CommandName,
CommandArgument );
OnCommand( e );
}

protected override int CreateChildControls( IEnumerable
dataSource, bool dataBinding )
{
//Controls.Clear();
CreateChildControls();

// TODO: return an int? (no. of rows
return 2;
}
}
}
.
```