NetAdvantage for ASP.NET Grid Feature Cheat Sheet

This guide is a feature-by-feature comparison of equivalent characteristics among the WebDataGrid™ and WebHierachicalDataGrid™ controls. Click on a feature to go directly to a code snippet.

Notes

- An instance of the `ScriptManager` component **is required** in all instances.
- This cheat sheet is an excellent resource for `UltraWebGrid` customers looking to move to the `WebDataGrid` or `WebHiearchicalDataGrid` controls.
- The notation of 'Same' for WebHierarchicalDataGrid implementations does not necessarily indicate that the required code is exactly the same. You may have to access the grid slightly differently, but generally the approach is the same.

Adding Behaviors to WebHierachicalDataGrid Child Bands

The easiest way to expose a behavior enabled on the parent grid to a child band of an instance of the WebHierarchicalDataGrid is to set the behavior's `EnableInheritance` property to `true` on the top-level grid. If, for some reason, you opt to not enable inheritance of a behavior there are a number of ways you may enable a specific behavior for a child band.

Often the procedure to add a behavior to the parent grid of a `WebHierarchicalDataGrid` control is the same as described for the `WebDataGrid` control associated to a behavior in this document. However, in order to enable behaviors on a non-inherited child band, you must add the behavior directly to the band via the designer, in markup or in code.

The following code snippets demonstrate how to explicitly add the `ColumnMoving` behavior a child band, although **you may add any grid behavior using this approach**.

In the Designer

1. **Enable Behavior on Child Band**
   1.1. Open the **Smart Tag**
   1.2. Select **Edit Bands**
   1.3. Select a band
   1.4. Expand **Behaviors** section
   1.5. Check **ColumnMoving**
   1.6. Configure in dialog window

In ASPX

```
<Bands>
    <ig:Band ...>
        <Columns>
            ...
        </Columns>
        <Behaviors>
            <ig:ColumnMoving />
        </Behaviors>
    </ig:Band>
</Bands>
```

In Code Behind

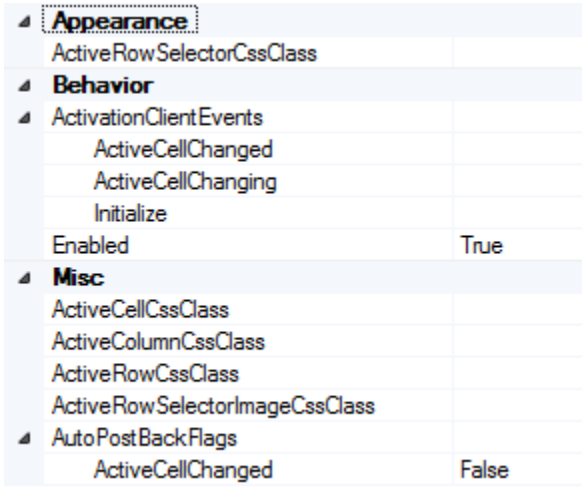
```
using Infragistics.Web.UI.GridControls;
...
protected void WebHierarchicalDataGrid1_InitializeBand(object sender, BandEventArgs e)
{
    e.Band.Behaviors.Add(new ColumnMoving());
}
```


Code Snippets

Activation

WebDataGrid

1. **Enable Activation Behavior**
   1.1. Open the **Smart Tag**
   1.2. Select **Edit Behaviors**
   1.3. Check **Activation**
   1.4. Configure in dialog window

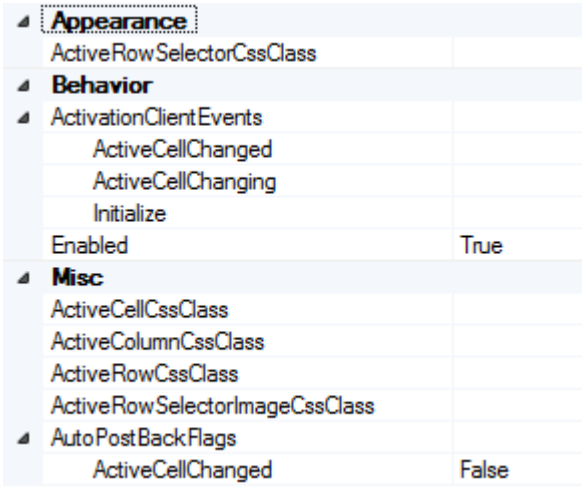


Setting Active Cell on the Client

```
function setActiveCell() {

    var grid = $find("<%= this.WebDataGrid1.ClientID %>");
    var behaviors = grid.get_behaviors();
    var activation = behaviors.get_activation();

    var rows = grid.get_rows();
    var row = rows.get_row(0);
    var cell = row.get_cell(1);

    activation.set_activeCell(cell);
```

WebHierarchicalDataGrid

1. **Enable Activation Behavior**
   1.1. Open the **Smart Tag**
   1.2. Select **Edit Behaviors**
   1.3. Check **Activation**
   1.4. Configure in dialog window



Setting Active Cell on the Client – Parent Level

```
function setActiveCellTopLevel() {

    var grid = $find("WebHierarchicalDataGrid1").get_gridView();
    var behaviors = grid.get_behaviors();
    var activation = behaviors.get_activation();

    var rows = grid.get_rows();
    var row = rows.get_row(0);
    var cell = row.get_cell(1);

    activation.set_activeCell(cell);
}
```

```
}
```

Getting Active Cell on the Client

```
function getGetCell() {

    var grid = $find("<%= this.WebDataGrid1.ClientID %>");
    var behaviors = grid.get_behaviors();
    var activation = behaviors.get_activation();

    return activation.get_activeCell();
}
```

More Info

- WebDataGrid Activation

Setting Active Cell on the Client – Parent Level

```
function setActiveCellRowIsland() {

    var grid = $find("WebHierarchicalDataGrid1");
    var parentGrid = grid.get_gridView();

    if (parentGrid != null) {

        var rows = parentGrid.get_rows();
        var row = rows.get_row(0);
        var cell = row.get_cell(2);

        var childGrid = row.get_rowIslands(0)[0];

        if (childGrid != null) {

            var behaviors = childGrid.get_behaviors();
            var activation = behaviors.get_activation()

            var cRows = childGrid.get_rows();
            var cRow = cRows.get_row(0);
            var cCell = cRow.get_cell(0);

            activation.set_activeCell(cCell);
        }
    }
}
```

Getting Active Cell on the Client – Parent Level

```
function getActiveCellTopLevel() {

    var grid = $find("WebHierarchicalDataGrid1").get_gridView();
    var behaviors = grid.get_behaviors();
    var activation = behaviors.get_activation();

    return activation.get_activeCell();
}
```

Getting Active Cell on the Client – Row Island Level

```
function getActiveCellRowIsland() {

    var grid = $find("WebHierarchicalDataGrid1");
    var parentGrid = grid.get_gridView();
    var activeCell;
```

```
    if (parentGrid != null) {

        var rows = parentGrid.get_rows();
        var row = rows.get_row(0);

        var childGrid = row.get_rowIslands(0)[0];

        if (childGrid != null) {

            var behaviors = childGrid.get_behaviors();
            var activation = behaviors.get_activation()

            activeCell = activation.get_activeCell();
        }
    }

    return activeCell;
}
```

More Info

- WebHierarchicalDataGrid Activation

Ajax: Events

WebDataGrid	WebHierarchicalDataGrid

1. Create client handlers for Ajax events
1.1. Open the **Properties** window
1.2. Expand the **Client Events** group
1.3. Create a handler for `AJAXResponse` and `AJAXResponseError`
1.3.1. Generate the handler by clicking on the drop down and assigning a name to the function
 	
The following example depicts how your functions may look:
 	
`function onAjaxResponseError(sender, eventArgs)`
`{`
`    // do something on the client`
`    // in response to the Ajax error`
`}`
	← Same


```
function onAjaxResponse(sender, eventArgs) {
    // do something on the client
    // in response to the successful Ajax request
}
```

Ajax: Load on Demand (Automatic)

WebDataGrid	WebHierarchicalDataGrid
Not Implemented	**Note:** See *Automatic Load on Demand* section for basic settings. When the grid's `EnableAjax` property is set to `True` (default value), data interactions are asynchronous. If the property is set to `False`, data interactions result in a full page postback.

Ajax: Load on Demand (Manual)

WebDataGrid	WebHierarchicalDataGrid

Not Implemented	Achieving Load on Demand requires you to first bind to the top-level data source in `Page_Load`, and then the child-level data on during `RowIslandsPopulating` event.
	
	`protected void Page_Load(object sender, EventArgs e)`
	`{`
	`  this.WebHierarchicalDataGrid1.DataSource =`
	`  this._productRepository.GetAll();`
	`}`
	
	`protected void WebHierarchicalDataGrid1_RowIslandsPopulating(object sender, Infragistics.Web.UI.GridControls.ContainerRowCancelEventArgs e)`
	`{`
	`  e.Cancel = true;`
	
	`  ContainerGrid child = new ContainerGrid();`
	
	`  // further configure container grid`
	`  // by customizing columns, behaviors, etc.`
	`  child.InitializeRow += new InitializeRowEventHandler(child_InitializeRow);`
	`  e.Row.RowIslands.Add(child);`


```
    int productID = Convert.ToInt32(e.Row.DataKey[0]);
    child.DataSource =
        this._supplierRepository.GetByProductID(productID);
    child.DataBind();
}
```

Ajax: Loading Indicator

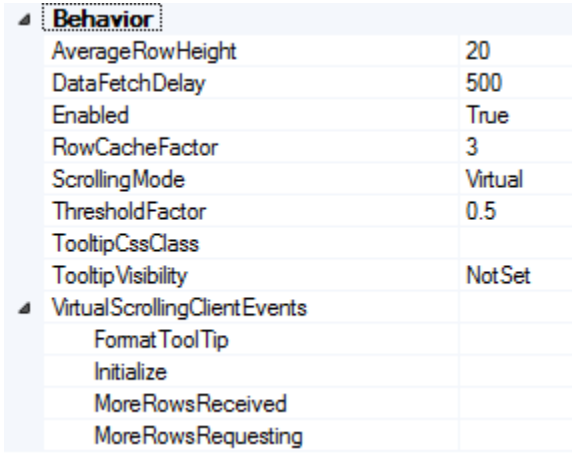
WebDataGrid	WebHierarchicalDataGrid
1. Configure the AjaxIndicator properties group 1.1. Open the grid's properties in the **Property window** 1.2. Expand the **AjaxIndicator** section 1.2.1. Set `ImageUrl` equal to the path of the loading image 1.2.2. Set `Location` equal to the location on the page of the indicator 1.2.3. Set `RelativeToControl` to determine if the location is relative to the grid or the page	← Same

AjaxIndicator	**(Has Data)**
AltText	Async post
BlockArea	NotSet
BlockCssClass	
CssClass	
Enabled	NotSet
FadeInDuration	
FadeInEquationType	EaseInOut
FadeOutDuration	
FadeOutEquationType	EaseInOut
ImageUrl	**~/ajax-loader.gif**
Location	**MiddleCenter**
OffsetLeft	
OffsetTop	
RelativeToControl	**True**
Text	

Note: The grid must have the `EnableAjax` property set to `True` for the

loading indicator to work properly.	

Ajax: Virtual Scrolling

WebDataGrid	WebHierarchicalDataGrid
1. Enable Virtual Scrolling Behavior 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check the **Virtual Scrolling Behavior** 1.4. Configure in dialog window 	Not Implemented

Automatic Load on Demand

WebDataGrid	WebHierarchicalDataGrid
Not Implemented	Set the `InitialDataBindDepth` property to the data levels desired during initial load. The default is `0` which only loads the root data. A value of `-1` disables load on demand and loads all data levels. **Note:** See the *Ajax: Load on Demand (Automatic)* section for Ajax interactions

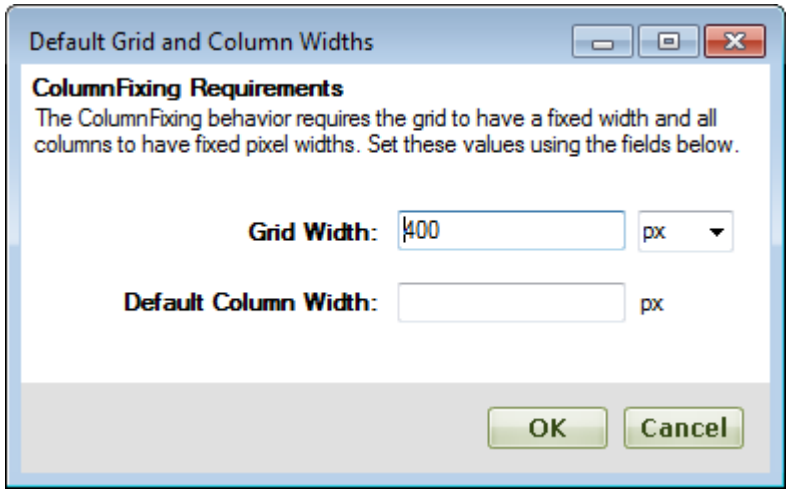
Columns: Hidden Columns

WebDataGrid	WebHierarchicalDataGrid
1. Configure Column Settings for hidden column 1.1. Open the **Smart Tag** and select **Edit Columns** 1.2. Choose a column from **Selected Fields** 1.3. Set: `Hidden = True`	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

Columns: Moving

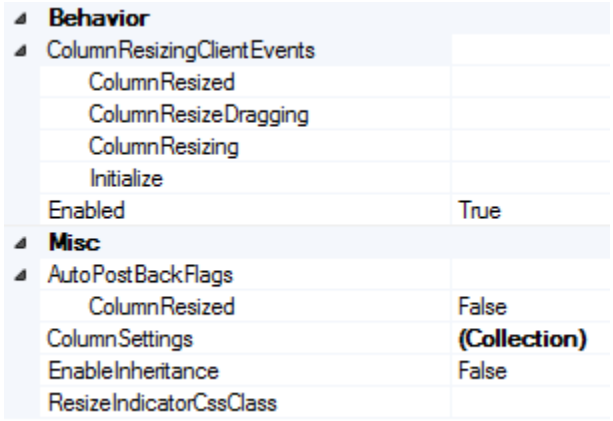
WebDataGrid	WebHierarchicalDataGrid
1. Enable Column Fixing Behavior 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Column Moving** 1.4. Configure in dialog window **Behavior** ColumnMovingClientEvents HeaderDragEnd HeaderDragStart HeaderDropped HeaderMove Initialize Enabled: True **Misc** BottomDragIndicatorCssClass ColumnSettings: **(Collection)** DragMarkupCssClass DragStyle: Slide EnableInheritance: False MiddleDragIndicatorCssClass TopDragIndicatorCssClass **More Info** • WebDataGrid Column Moving	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

Columns: Pinned

WebDataGrid	WebHierarchicalDataGrid
1. Enable Column Fixing Behavior 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Column Fixing** 1.4. Designate Default Column width  **2. Further configure the Column Fixing Behavior in the dialog window (if necessary)** **Note:** To designate the location of the pinned columns set the `FixLocation` to either `Left` or `Right` in the Fixing Behavior properties.	Not Implemented

Columns: Resizing

WebDataGrid	WebHierarchicalDataGrid
1. **Enable Column Resizing Behavior**	← Same for parent grid

1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Column Resizing** 1.4. Configure in dialog window 	To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

Columns: Unbound Columns

WebDataGrid	WebHierarchicalDataGrid
1. **Create an UnboundField** 1.1. Open the **Smart Tag** 1.2. Select **Edit Columns** 1.3. Select **UnboundField** 1.4. Click the **Add Field** button 1.5. Select the new field in the **Selected Fields** list 1.5.1. Set `DataFormatString` to `{0:c2}` (your value will vary depending on needs) 1.5.2. Set `DataType` to `System.Decimal` (your value will vary depending on needs) 1.5.3. Set the `Key` field to a unique value 1.5.4. Set `Header` -> `Text` to the desired header text 1.6. Close the dialog 2. **Implement the field's logic** in the `RowInitialized` event 2.1. In the grid's **Properties** window, switch to the **Events** view	← Approach for the `WebHierarchicalDataGrid` is generally the same, although configuring for child bands requires you to add settings and apply the logic in scope of a child band.

2.2. Double-click on the **RowInitialized** event
2.3. Implement the logic for your unbound field in the `RowInitialized` event handler

Creating an Unbound Column

The following code implements the logic for a total column which multiplies price by quantity.

UnitPrice	UnitsInStock	Total Retail
18.0000	39	$702.00

In ASPX

```
<Columns>
    ...
    <ig:UnboundField
        Key="TotalRetail"
        DataType="System.Decimal"
        DataFormatString="{0:c2}">
            <Header Text="Total Retail" />
    </ig:UnboundField>
</Columns>
```

In Code Behind

Execute in `InitializeRow` event handler

```
using Infragistics.Web.UI.GridControls;
...
GridRecordItem unbound;
GridRecordItem bound1;
GridRecordItem bound2;

int indexOfUnbound = 4;
int indexOfBound1 = 3;
int indexOfBound2 = 2;

unbound = e.Row.Items[indexOfUnbound];
bound1 = e.Row.Items[indexOfBound1];
bound2 = e.Row.Items[indexOfBound2];

decimal unitPrice = Convert.ToDecimal(bound1.Value);
```

```
int qty = Convert.ToInt32(bound2.Value);
decimal total = unitPrice * qty;

e.Row.Items[indexOfUnbound].Value = total;
```

Columns: Checkbox Columns

WebDataGrid	WebHierarchicalDataGrid

Binding to a `boolean` field in the grid will create a `BoundCheckboxField` by default as the column type.
 	
Additionally, you may display any bound data as a checkbox field by implementing the `IBooleanConverter` interface.
 	
Binding Non-Boolean Data as Checkboxes
In ASPX
`<Columns>`
`<ig:BoundCheckBoxField`
`DataFieldName="UnitPrice"`
`CellText=" High Unit Price"`
`Key="UnitPrice">`
`<Header Text="UnitPrice" />`
`<CheckBox`
`CheckedImageUrl="~/images/checked.png"`
`UncheckedImageUrl="~/images/unchecked.png"`
`PartialImageUrl="~/images/partial.png" />`
`</ig:BoundCheckBoxField>`
`</Columns>`
 	
In Code Behind
`public partial class WDG : System.Web.UI.Page`
`{`
`private int UNIT_PRICE_COLUMN_INDEX = 0;`
 	
`protected void Page_Load(object sender, EventArgs e)`
`{`
`BoundCheckBoxField field = (BoundCheckBoxField)`
`this.WebDataGrid1.Columns[UNIT_PRICE_COLUMN_INDEX];`
 	
`field.ValueConverter = new UnitPriceConverter();`
`}`
	← Approach for the `WebHierarchicalDataGrid` is generally the same, although configuring for child bands requires you to add settings and assign the converter in scope of a child band.


```
}

public class UnitPriceConverter :
Infragistics.Web.UI.GridControls.IBooleanConverter
{
    public object DefaultFalseValue
    {
        get { return 0; }
    }

    public object DefaultTrueValue
    {
        get { return 100; }
    }

    public bool IsFalse(object value)
    {
        if (value != null && value is decimal)
        {
            return (decimal)value < 100;
        }
        return false;
    }

    public bool IsTrue(object value)
    {
        if (value != null && value is decimal)
        {
            return (decimal)value >= 100;
        }
        return false;
    }
}
```

Note: Your implementation will likely require support for other data types beyond a `decimal` type.

Columns: Summary Rows

	WebHierarchicalDataGrid
1. **Add Summary Row Behavior** 1.1. Open the **Smart Tag**	← Same for parent grid

1.2. Select **Edit Behaviors** 1.3. Check **Summary Row** **Note:** You can compact summaries by showing different summary details in the same row by setting `CompactRendering` to `On`.	To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

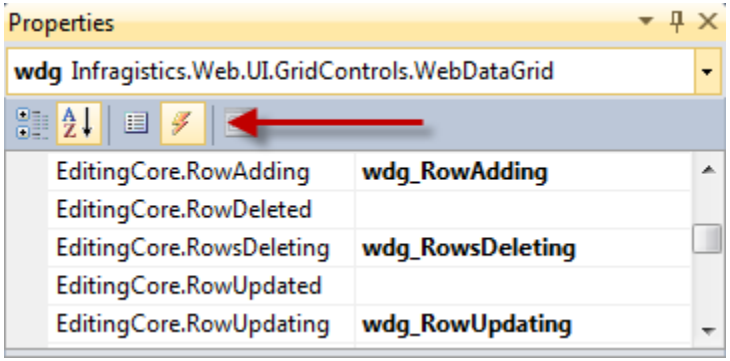
CRUD: Auto

WebDataGrid	WebHierarchicalDataGrid
CRUD actions for the grid are batched, therefore once your set up Auto CRUD you still need to add a control to the page to send change message back to the server. **1. Setup the data source** 1.1. Bind the grid to a data source control that implements functionality for inserting, selecting, updating and deleting **2. Set the DataKeyFields property** 2.1. From the **Properties** window set the `DataKeyFields` property the field name of the primary key value(s) **3. Enable editing behaviors** 3.1. Open the **Smart Tag** 3.2. Select **Edit Behaviors** 3.3. Check **Selection** 3.3.1.Set `RowSelectType` equal to `Single` 3.4. Check **Row Selectors** 3.5. Check **Editing Core** 3.5.1.Ensure `AutoCRUD` is equal to `True` 3.6. Check **Cell Editing** 3.7. Check **Row Adding** 3.8. Check **Row Deleting** 3.9. Click **OK** 4. **Add Control to Cause Postback** 4.1. Add a control like a server Button to initiate postback to the server	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

More Info • WebDataGrid Auto CRUD (video)	

CRUD: Manual

WebDataGrid	WebHierarchicalDataGrid
1. Configure grid for manual CRUD operations 1.1. From the **Properties** window set the `DataKeyFields` property the field name of the primary key value(s) 1.2. Set the `EnableAjax` property equal to `False` **2. Enable CRUD behaviors** **2.1.** Open the **Smart Tag** 2.2. Select **Edit Behaviors** 2.3. Check **Selection** 2.4. Check **Row Selection** 2.5. Check **Activation** 2.6. Check **Editing Core** 2.6.1.Ensure `AutoCRUD` is set to `False` 2.7. Check **Cell Editing** 2.7.1.Set `EditModeActions` -> `EnableOnKeyPress` equal to `True` 2.8. Check **Row Adding** 2.8.1.Set `EditModeActions` -> `EnableOnKeyPress` equal to `True` 2.9. Check **Row Deleting** **3. Subscribe to CRUD events** 3.1. From the Properties window, double click on these events to create event handlers: 3.1.1.**EditingCore.RowAdding** 3.1.2. **EditingCore.RowDeleting** 3.1.3. **EditingCore.RowUpdating**	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.



4. Implement CRUD logic

4.1. In general terms you must implement logic that facilitates the CRUD messages as well as expose the latest data to the grid. One approach you may use it to re-bind the grid on each page load and allow the event handlers to carry out the CRUD operations. The code snippets below reflect this design.

```
using Infragistics.Web.UI.GridControls;
…
public void BindData()
{
    this.wdg.Rows.Clear();
    this.wdg.DataSource = this.GetData();
    this.wdg.DataBind();
}

public IList<Person> GetData()
{
    // make a call to data access
    // layer to retrieve data

    // the return type does not have to be an IList
    // nor of type Person

    // the method signature is for placholder purposes only
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!this.Page.IsPostBack)
```



```
    {
        this.BindData();
    }
}

protected void wdg_RowAdding(object sender,
Infragistics.Web.UI.GridControls.RowAddingEventArgs e)
{
    Person person = new Person()
    {
        FirstName = e.Values["FirstName"].ToString(),
        LastName = e.Values["LastName"].ToString()
    };

    this.Repository.Insert(person);
    this.BindData();
}

protected void wdg_RowsDeleting(object sender, RowDeletingEventArgs e)
{
    int id = Convert.ToInt32(e.Row.DataKey[0]);
    this.Repository.Delete(id);
    this.BindData();
}

protected void wdg_RowUpdating(object sender,
Infragistics.Web.UI.GridControls.RowUpdatingEventArgs e)
{
    Person person = new Person()
    {
        Id = Convert.ToInt32(e.Row.DataKey[0]),
        FirstName = e.Values["FirstName"].ToString(),
        LastName = e.Values["LastName"].ToString()
    };

    this.Repository.Update(person);
    this.BindData();
}
```

DataBinding: HeirachicalDataSource

WebDataGrid	WebHierarchicalDataGrid
Not Implemented	The `WebHierarchicalDataSource` control acts as a wrapper and connector for

two or more data source controls that correlate together via related fields.

The following procedure describes how to add two data sources as child views to the `WebHierarchicalDataSource` control.

1. **Add a WebHierarchicalDataGrid to the page**
2. **Add and configure a WebHierarchicalDataSource control on the page**
   2.1. Add a `WebHierarchicalDataSource` control to the page
   2.2. Set the `ID` property of the control to desired value
   2.3. Open the data source's **Smart Tag**
   2.4. Select **Configure DataSource**
   2.5. Click **Add View**
3. **Add First Data Source**
   3.1. From the **DataSource** drop down, select **New Data Source**
   3.2. Select a data source type from the dialog window
      3.2.1. Configure data source via data source configuration wizard and click **Finish**

   Note: Make sure the data source includes selection of the foreign key field which will relate to the primary key value of the related data added in step 4.3.1.

   3.3. Click **OK** in the *WebHierarchicalDataSource Designer* dialog window
4. **Add Second Data Source**
   4.1. Click **Add Child** under the data source you just created in the *Edit WebHierarchicalDataSource* dialog window
   4.2. Select **New Data Source** from the **Child Data Source** drop down
   4.3. Select a data source type from the dialog window
      4.3.1. Configure data source via data source configuration wizard and click **Finish**

   Note: Make sure the data source includes selection of the primary key field which will relate to the foreign key value of the related data added in step 3.2.1.

5. **Relate Fields Among Data Sources in *Edit WebHierarchicalDataSource* dialog window**
   5.1. Click on the **Parent Columns** drop down

	5.1.1. Select the **primary key field** from the field options 5.2. Click on the Child Columns drop down 5.2.1. Select the **foreign key field** from the field options 5.3. Click **OK** **6. Close the Designer Dialog** 6.1. Click **OK** **7. Associate the WebHierarchicalDataSource to the WebHierarchicalDataGrid** 7.1. Open the grid's **Smart Tag** 7.2. Click on the drop down for **Choose Data Source** 7.3. Select the ID of the WebHierarchicalDataSource as defined in step 2.2 **Note:** You may see a dialog box asking you if you want to regenerate the column and key settings for the grid. If you are setting up the grid for the first time then you may safely click **Yes**. If you are unsure, you may want to click **No** and manually configure the column and key options. **More Info** • Getting Started with WebHierarchicalDataSource

DataBinding: DataSet

WebDataGrid	WebHierarchicalDataGrid

To bind a `DataSet` to the `WebDataGrid`, set the `DataSource` property equal to a `DataSet` and then call `DataBind` on the grid.
 	
In Code Behind
`protected void Page_Load(object sender, EventArgs e)`
`{`
`  ProductRepository repository = new ProductRepository();`
`  DataSet ds = repository.GetAllDataSet();`
 	
`  this.WebDataGrid1.DataSource = ds;`
`  this.WebDataGrid1.DataBind();`
`}`
	← Same

DataBinding: DataTable

WebDataGrid	WebHierarchicalDataGrid

To bind a `DataTable` to the `WebDataGrid`, set the `DataSource` property equal to a `DataTable` and then call `DataBind` on the grid.
 	
In Code Behind
`protected void Page_Load(object sender, EventArgs e)`
`{`
`  ProductRepository repository = new ProductRepository();`
`  DataTable ds = repository.GetAllDataTable();`
 	
`  this.WebDataGrid1.DataSource = ds;`
`  this.WebDataGrid1.DataBind();`
`}`
	The `WebHierarchicalDataGrid` is able to bind to a `DataTable` in the same manner as the `WebDataGrid`, but you must first add the `DataTable` to a `DataSet`.

DataBinding: IEnumerable

WebDataGrid	WebHierarchicalDataGrid

To bind `IEnumerable` collections to the `WebDataGrid`, set the `DataSource` property equal to an `IEnumerable` collection and then call `DataBind` on the grid.
 	
In Code Behind
`protected void Page_Load(object sender, EventArgs e)`
`{`
`  ProductRepository repository = new ProductRepository();`
`  IEnumerable<Product> data = repository.GetAllEnumerable();`
`  this.WebDataGrid1.DataSource = data;`
`  this.WebDataGrid1.DataBind();`
`}`
	← Same

DataBinding: AccessDataSource

WebDataGrid	WebHierarchicalDataGrid
1. Add a AccessDataSource control to the page 1.1. Drag a **AccessDataSource** control to the design surface 1.2. Open the Properties window and give the control a value for the `Id` property	Follow the procedure as described in *DataBinding: HierarchicalDataSource* and select `AccessDataSource` as the data source type for steps 3.2.1 and/or 4.3.1.

1.3. Open the data source's **Smart Tag** 1.4. Follow the wizard to complete configuration of the data source control **2. Associate the data source control to the grid** 2.1. Open the grid's **Smart Tag** 2.2. Click on the drop down next to the label **Choose Data Source** 2.3. Select the data source **ID** as defined in step 1.2 **Note:** You may see a dialog box asking you if you want to regenerate the column and key settings for the grid. If you are setting up the grid for the first time then you may safely click **Yes**. If you are unsure, you may want to click **No** and manually configure the column and key options.	

DataBinding: SqlDataSource

WebDataGrid	WebHierarchicalDataGrid
1. Add a SqlDataSource control to the page 1.1. Drag a **SqlDataSource** control to the design surface 1.2. Open the Properties window and give the control a value for the `Id` property 1.3. Open the data source's **Smart Tag** 1.4. Follow the wizard to complete configuration of the data source control **2. Associate the data source control to the grid** 2.1. Open the grid's **Smart Tag** 2.2. Click on the drop down next to the label **Choose Data Source** 2.3. Select the data source **ID** as defined in step 1.2 **Note:** You may see a dialog box asking you if you want to regenerate the column and key settings for the grid. If you are setting up the grid for the first time then you may safely click **Yes**. If you are unsure, you may want to click **No** and manually configure the column and key options.	Follow the procedure as described in *DataBinding: HierarchicalDataSource* and select `SqlDataSource` as the data source type for steps 3.2.1 and/or 4.3.1.

DataBinding: ObjectDataSource

WebDataGrid	WebHierarchicalDataGrid
1. Add a ObjectDataSource control to the page 1.1. Drag a **ObjectDataSource** control to the design surface 1.2. Open the Properties window and give the control a value for the `Id` property 1.3. Open the data source's **Smart Tag** 1.4. Follow the wizard to complete configuration of the data source control **2. Associate the data source control to the grid** 2.1. Open the grid's **Smart Tag** 2.2. Click on the drop down next to the label **Choose Data Source** 2.3. Select the data source **ID** as defined in step 1.2 **Note:** You may see a dialog box asking you if you want to regenerate the column and key settings for the grid. If you are setting up the grid for the first time then you may safely click **Yes**. If you are unsure, you may want to click **No** and manually configure the column and key options.	Follow the procedure as described in *DataBinding: Hierarchical Data* and select `ObjectDataSource` as the data source type for steps 3.2.1 and/or 4.3.1.

DataBinding: LinqDataSource

WebDataGrid	WebHierarchicalDataGrid
1. Add a LinqDataSource control to the page 1.1. Drag a **LinqDataSource** control to the design surface 1.2. Open the Properties window and give the control a value for the `Id` property 1.3. Open the data source's **Smart Tag** 1.4. Follow the wizard to complete configuration of the data source control **2. Associate the data source control to the grid** 2.1. Open the grid's **Smart Tag** 2.2. Click on the drop down next to the label **Choose Data Source** 2.3. Select the data source **ID** as defined in step 1.2	Follow the procedure as described in *DataBinding: Hierarchical Data* and select `LinqDataSource` as the data source type for steps 3.2.1 and/or 4.3.1.

Note: You may see a dialog box asking you if you want to regenerate the column and key settings for the grid. If you are setting up the grid for the first time then you may safely click **Yes**. If you are unsure, you may want to click **No** and manually configure the column and key options.

Editing: Adding Rows

WebDataGrid

1. **Setup the data source**
   1.1. Bind the grid to a data source control that implements selection and inserting.
2. **Set the DataKeyFields property**
   2.1. From the **Properties** window set the `DataKeyFields` property the field name of the primary key value(s)
3. **Enable editing behaviors**
   3.1. Open the **Smart Tag**
   3.2. Select **Edit Behaviors**
   3.3. Check **Editing Core**

On the Client

```
function addRow() {
    var grid = $find("<%= this.WebDataGrid1.ClientID %>");

    // Fill array with all values to insert into the row
    var values = ["10000", "New Product"];

    grid.get_rows().add(values);
}
```

On the Server

To add rows to the grid on the server, add the data to your data source and re-bind the grid.

Customizations

WebHierarchicalDataGrid

1. **Setup the data source**
   1.1. Bind the grid to a data source control that implements selection and inserting.
2. **Set the DataKeyFields property**
   2.1. From the **Properties** window set the `DataKeyFields` property the field name of the primary key value(s)
3. **Enable editing behaviors**
   3.1. Open the **Smart Tag**
   3.2. Select **Edit Behaviors**
   3.3. Check **Editing Core**

On the Client

Root Level

```
var grid = $find("WebHierarchicalDataGrid1");

var topRowIsland = grid.get_gridView();
var cellValues = ["1", "Bob", "Green", "1/2/1983"];
topRowIsland.get_rows().add(cellValues);
```

Child Level

```
var grid = $find("WebHierarchicalDataGrid1");

var childGrid = topRowIsland.get_rows().get_row(3).get_rowIslands()[0];
cellValues = ["1", "25 Main Road", "New York", "NY", "19234"];
childGrid.get_rows().add(cellValues);
```

On the Server

You can further customize the process of adding new records on a column-by-column basis by providing values for `DefaultValueAsString`, `ReadOnly`, `EditorID` and `ValidatorID` on `ColumnSettings`.

In ASPX

```
<Behaviors>
    <ig:EditingCore>
        <Behaviors>
            <ig:RowAdding>
                <ColumnSettings>
                    <%-- Setting a default value for new data --%>
                    <ig:RowAddingColumnSetting
                        ColumnKey="Size"
                        DefaultValueAsString="3" />

                    <%-- Setting “Id” column to read-only, therefore
                    a value cannot be provided --%>
                    <ig:RowAddingColumnSetting
                        ColumnKey="Id" ReadOnly="true" />

                    <%-- Setting up an editor & validator --%>
                    <ig:RowAddingColumnSetting
                        ColumnKey="OrderDate"
                        EditorID="DateTimePicker1"
                        ValidatorID="myValidator1" />
                </ColumnSettings>
            </ig:RowAdding>
        </Behaviors>
    </ig:EditingCore>
</Behaviors>
```

In Code Behind

Execute in the `Page_Load` event handler

```
RowAdding rowAddingBehavior =
this.WebDataGrid1.Behaviors.EditingCore.Behaviors.RowAdding;

/* Setting a default value for new data */
RowAddingColumnSetting sizeSetting = new RowAddingColumnSetting();
sizeSetting.ColumnKey = "Size";
sizeSetting.DefaultValueAsString = "3";
rowAddingBehavior.ColumnSettings.Add(sizeSetting);

/* Setting “Id” column as read only, therefore a value cannot be
```

To add rows to the grid on the server, add the data to your data source and re-bind the grid.

Customizations

You can further customize the process of adding new records on a column-by-column basis by providing values for `DefaultValueAsString`, `ReadOnly`, `EditorID` and `ValidatorID` on `ColumnSettings`.

In ASPX

← The markup is basically the same for the WebHierarchicalDataGrid as it is for the `WebDataGrid`, you just need to customize the columns at the appropriate level in the grid.

In Code Behind

Execute in the `InitializeBand` event handler

```
if (e.Band.DataMember == "Root")
{
    RowAdding rowAddingBehavior =
        e.Band.Behaviors.EditingCore.Behaviors.RowAdding;

    /* Setting a default value for the new data */
    RowAddingColumnSetting sizeSetting = new RowAddingColumnSetting();
    sizeSetting.ColumnKey = "Size";
    sizeSetting.DefaultValueAsString = "3";
    rowAddingBehavior.ColumnSettings.Add(sizeSetting);

    /* Setting “Id” column as read only, therefore a value cannot be
provided */
    RowAddingColumnSetting idSetting = new RowAddingColumnSetting();
    idSetting.ColumnKey = "Id";
    idSetting.ReadOnly = true;
    rowAddingBehavior.ColumnSettings.Add(idSetting);

    /* Setting up an editor & validator */
    RowAddingColumnSetting orderDateSetting =
        new RowAddingColumnSetting();
    orderDateSetting.ColumnKey = "OrderDate";
    orderDateSetting.EditorID = "DateTimePicker1";
    orderDateSetting.ValidatorID = "myValidator1";
    rowAddingBehavior.ColumnSettings.Add(orderDateSetting);
}
```



```
provided */
RowAddingColumnSetting idSetting = new RowAddingColumnSetting();
idSetting.ColumnKey = "Id";
idSetting.ReadOnly = true;
rowAddingBehavior.ColumnSettings.Add(idSetting);

/* Setting up an editor & validator */
RowAddingColumnSetting orderDateSetting = new RowAddingColumnSetting();
orderDateSetting.ColumnKey = "OrderDate";
orderDateSetting.EditorID = "DateTimePicker1";
orderDateSetting.ValidatorID = "myValidator1";
rowAddingBehavior.ColumnSettings.Add(orderDateSetting);
```

```
else if (e.Band.DataMember == "SecondLevel")
{
    RowAdding rowAddingBehavior =
        e.Band.Behaviors.EditingCore.Behaviors.RowAdding;
    /* Configuring child grids would use the same approach
       as shown above. */
}
```

Editing: Deleting Rows

WebDataGrid

1. **Add the Row Deleting behavior**
   1.1. Open the **Smart Tag**
   1.2. Click **Edit Behaviors**
   1.3. Check **Selection**
   1.4. Check **Editing Core**
   1.5. Check **Row Deleting**

Once the deleting behavior is added to the grid then you may remove rows from the grid a number of different ways.

Note: If the data source attached to the grid is not automatically updated by the grid then you must handle the update events manually to make changes on the server.

On the Server
To remove a row from the grid on the server, remove the row from the data source and rebind the grid.

On the Client
Delete Selected Rows - Basic

WebHierarchicalDataGrid

1. **Add the Row Deleting behavior**
   1.1. Open the **Smart Tag**
   1.2. Click **Edit Behaviors**
   1.3. Check **Selection**
   1.4. Check **Editing Core**
   1.5. Check **Row Deleting**

Once the deleting behavior is added to the grid then you may remove rows from the grid a number of different ways.

Note: If the data source attached to the grid is not automatically updated by the grid then you must handle the update events manually to make changes on the server.

On the Server
To remove a row from the grid on the server, remove the row from the data source and rebind the grid.

On the Client
Delete Selected Rows at Any Level


```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var gridRows = grid.get_rows()

var selectedRows = grid.get_behaviors().
                        get_selection().
                        get_selectedRows();

var rows = new Array();
var i1 = 0;

for (var i = selectedRows.get_length() - 1; i >= 0; i--) {

        rows[i1] = selectedRows.getItem(i);
      i1++;
}

grid.get_behaviors().
        get_editingCore().
        get_behaviors().
        get_rowDeleting().
        deleteRows(rows);
```

Delete Selected Rows – Using 'Remove' Method

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var gridRows = grid.get_rows()

var selectedRows = grid.get_behaviors().
                        get_selection().
                        get_selectedRows();

for (var i = selectedRows.get_length() - 1; i >= 0; i--) {
        var row = selectedRows.getItem(i);
        gridRows.remove(row);
}
```

Delete non-Selected Rows

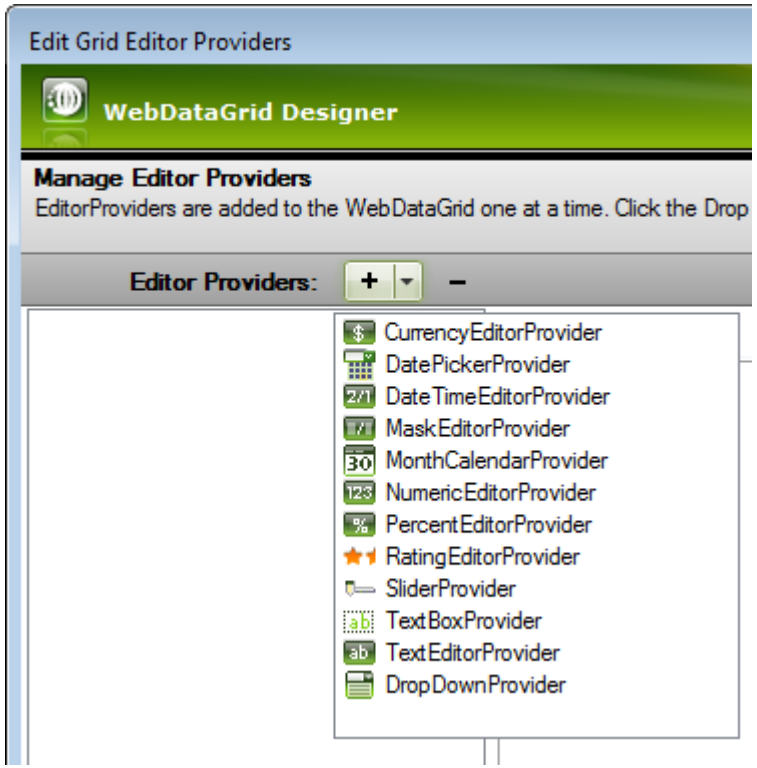
```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var row = grid.get_rows().get_row(0);

grid.get_rows().remove(row);
```

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var selection = grid.get_gridView().
                        get_behaviors().
                        get_selection();

// get_selectedRowsResolved() gets selected rows across all grids
// it returns a row array
var selectedRows = selection.get_selectedRowsResolved();

for (var i=0; i < selectedRows.length; i++) {
        var row = selectedRows[i];
        var containerGrid = row.get_grid();
        var gridRows = containerGrid.get_rows();
        gridRows.remove(row);
}
```

Delete non-Selected Rows

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var row;

// Remove first PARENT row
row = parentGrid.get_rows().get_row(0);
parentGrid.get_rows().remove(row);

var childGrid = grid.get_gridView().
                        get_rows().
                        get_row(0).
                        get_rowIslands(0)[0];

// Remove first CHILD row
row = childGrid.get_rows().get_row(0);
childGrid.get_rows().remove(row);
```

Editing: Embeddable Editors / Editor Providers

WebDataGrid	WebHierarchicalDataGrid
1. Add editor providers to the grid 1.1. Open the **Smart Tag** 1.2. Select **Edit Grid Editor Providers** 1.3. Click on the **+** button in the resulting dialog  1.4. Select the desired editor provider 1.5. Configure by providing the appropriate value for the **ID** property in the detail pane on the right side of the dialog. 1.6. Click **OK** **2. Associate an editor provider to a column** 2.1. Open the **Smart Tag**	← Same

2.2. Select **Edit Behaviors** 2.3. Check **Editing Core** 2.4. Check **Cell Editing** 2.5. Click the ellipsis button on the **ColumnSettings** property 2.6. Click the **add item** button 2.7. Select a value for the **ColumnKey** 2.8. Select a value for the **EditorID** 2.9. Click **OK** on all open dialogs	

Export to Excel

WebDataGrid	WebHierarchicalDataGrid

1. Add a Reference to Infragistics.Documents.Excel
2. Add a WebExcelExporter to the page
1.1. Drag the `WebExcelExporter` on the page
1.2. In the Properties window, set Id equal to eExporter
3. Create a button to handle the export procedure
2.1. Drag a Button onto the page
2.2. In the Properties window set the Id property equal to btnExport
2.3. Double-click on the button to create a click handler in the code behind
 	
In Code Behind
Execute in the `btnExport_Click` event handler
 	
`// file will get extension based upon workbook format`
`this.eExporter.DownloadName = "data";`
 	
`eExporter.WorkbookFormat =`
`Infragistics.Documents.Excel.WorkbookFormat.Excel2007;`
 	
`// export mode- all grid data or all visible data`
`eExporter.DataExportMode =`
`Infragistics.Web.UI.GridControls.DataExportMode.AllDataInDataSource;`
 	
`// determines whether to apply DataFormatString to values before assigning to excel cell`
`eExporter.DisableCellValueFormatting = true;`
	← Same


```
// determines if styles should be exported
eExporter.EnableStylesExport = true;

// how many rows to place between multiple grids exported
eExporter.GridsRowSpacing = 2;

eExporter.Export(new WebControl[] { WebDataGrid1, WebDataGrid2 });
```

Export to PDF

WebDataGrid	WebHierarchicalDataGrid
1. Add a WebDocumentExplorer to the page 1.1. Drag the `WebDocumentExporer` on the page 1.2. In the Properties window set `Id` equal to `dExporter` **2. Create a button to handle the export procedure** 2.1. Drag a `Button` on to the page 2.2. In the Properties window set the `Id` property equal to `btnExport` 2.3. Double-click on the button to create a click handler in the code behind **In Code Behind** Execute in the `btnExport_Click` event handler	← Same

```
this.dExporter.DownloadName = "data.pdf";

dExporter.Format = Infragistics.Web.UI.GridControls.FileFormat.PDF;

// export mode – all grid data or all visible data
dExporter.DataExportMode = Infragistics.
                                    Web.
                                    UI.
                                    GridControls.
                                    DataExportMode.
                                    DataInGridOnly;

dExporter.TargetPaperOrientation = Infragistics.
                                           Documents.
                                           Report.
                                           PageOrientation.
```



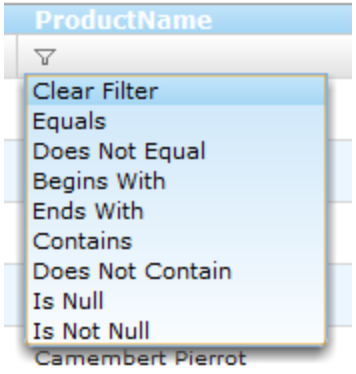
```
                                        Portrait;

dExporter.Margins = PageMargins.Normal;
dExporter.TargetPaperSize = PageSizes.A4;

// export the grids from the page
// if you want to export more than two grids
//   add them as WebControl[] array
dExporter.Export((WebDataGrid1, WebDataGrid2);
```

Filtering

WebDataGrid

1. **Add the Row Deleting behavior**
   1.1. Open the **Smart Tag**
   1.2. Click **Edit Behaviors**
   1.3. Check **Filtering**

Once the Filtering behavior is enabled on the grid the user may filter columns based on a number of different criteria:

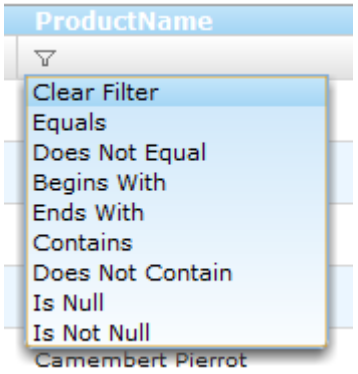


By default the Filtering behavior is enabled for all columns with blank filter rules. The following snippets demonstrate how to programmatically customize the filter settings.

WebHierarchicalDataGrid

1. **Add the Row Deleting behavior**
   1.1. Open the **Smart Tag**
   1.2. Click **Edit Behaviors**
   1.3. Check **Filtering**
      1.3.1. Set `EnableInheritance` equal to `True`

Once the Filtering behavior is enabled on the grid the user may filter columns based on a number of different criteria:



By default the Filtering behavior is enabled for all columns with blank filter rules. The following snippets demonstrate how to programmatically customize the filter settings.

Disable Filtering for a Single Column

ASPX

```
<Behaviors>
        <ig:Filtering>
                <ColumnSettings>
                        <ig:ColumnFilteringSetting
                                ColumnKey="ProductName"
                                Enabled="false" />
                </ColumnSettings>
        </ig:Filtering>
</Behaviors>
```

Code Behind

```
using Infragistics.Web.UI.GridControls;
...
ColumnFilteringSetting columnSetting = new
                    ColumnFilteringSetting(this.WebDataGrid1);
columnSetting.ColumnKey = "ProductName";
columnSetting.Enabled = false;

ColumnFilteringSettings settings = this.WebDataGrid1.
                                                Behaviors.
                                                Filtering.
                                                ColumnSettings;
settings.Add(columnSetting);
```

Predefined Column Filters

The snippets below show you how to implement a **text** filter, but the grid supports **boolean**, **date** and **number** filter rules as well.

ASPX

```
<ColumnFilters>
    <ig:ColumnFilter ColumnKey="ProductName">
        <ConditionWrapper>
            <ig:RuleTextNode Rule="Contains" Value="al" />
        </ConditionWrapper>
    </ig:ColumnFilter>
</ColumnFilters>
```

Code Behind

```
RuleTextNode textCondition = new RuleTextNode(TextFilterRules.Contains,
"al");
```

Disable Filtering for a Single Column

ASPX – Root Level

```
<Behaviors>
    <ig:Filtering EnableInheritance="True">
        <ColumnSettings>
            <ig:ColumnFilteringSetting
                    ColumnKey="CompanyName"
                    Enabled="False" />
        </ColumnSettings>
    </ig:Filtering>
</Behaviors>
```

ASPX – Band Level

```
<Bands>
    <ig:Band ...>
...
        <Behaviors>
            <ig:Filtering>
                <ColumnSettings>
                    <ig:ColumnFilteringSetting
                            ColumnKey="CompanyName"
                            Enabled="False" />
                </ColumnSettings>
            </ig:Filtering>
        </Behaviors>
    </ig:Band>
</Bands>
```

Code Behind – Root Level

Execute in the `Page_Load` event handler

```
ColumnFilteringSetting columnSetting = new
ColumnFilteringSetting(this.WebHierarchicalDataGrid1.GridView);
columnSetting.ColumnKey = "ProductName";
columnSetting.Enabled = false;

this.WebHierarchicalDataGrid1.
         GridView.
         Behaviors.
         Filtering.
         ColumnSettings.
         Add(columnSetting);
```

Code Behind – Band Level


```
ColumnFilter columnFilter = new ColumnFilter();
columnFilter.ColumnKey = "ProductName";
columnFilter.Condition = textCondition;

this.grid.Behaviors.Filtering.ColumnFilters.Add(columnFilter);

this.WebDataGrid1.Behaviors.Filtering.ApplyFilter();
```

In JavaScript

```
var grid = $find("<%= this.WebDataGrid1.ClientID %>");

var columnFilter = grid.
                        get_behaviors().
                        get_filtering().
                        create_columnFilter("ProductName");

var condition2 = columnFilter2.get_condition();
condition2.set_rule($IG.TextFilterRules.Contains);
condition2.set_value("al");

var columnFilters = new Array(columnFilter);

grid.get_behaviors().
    get_filtering().
    add_columnFilterRange(columnFilters);

grid.get_behaviors().get_filtering().applyFilters();
```

Execute in the `InitializeBand` event hander.

```
ColumnFilteringSetting settings = new ColumnFilteringSetting();
settings.ColumnKey = "SupplierID";
settings.Enabled = false;
e.Band.Behaviors.Filtering.ColumnSettings.Add(settings);
```

Code Behind – Row Island Level

Execute in the `RowIslandCreated` event handler

```
ColumnFilteringSetting columnSetting = new
ColumnFilteringSetting(e.RowIsland);
columnSetting.ColumnKey = "SupplierID";
columnSetting.Enabled = false;
e.RowIsland.Behaviors.Filtering.ColumnSettings.Add(columnSetting);
```

In JavaScript – Root Level

In JavaScript – Band Level

Predefined Column Filters

ASPX – Root Level

```
<Behaviors>
    <ig:Filtering EnableInheritance="True">
        <ColumnFilters>
            <ig:ColumnFilter ColumnKey="ProductName">
                <ConditionWrapper>
                    <ig:RuleTextNode Rule="Contains" Value="al" />
                </ConditionWrapper>
            </ig:ColumnFilter>
        </ColumnFilters>
    </ig:Filtering>
</Behaviors>
```

ASPX – Band Level

```
<Bands>
    <ig:Band ...>
        <Behaviors>
            <ColumnFilters>
                <ig:ColumnFilter ColumnKey="CompanyName">
                    <ConditionWrapper>
                        <ig:RuleTextNode
                                    Rule="Contains"
```



```
                                        Value="al" />
                    </ConditionWrapper>
                </ig:ColumnFilter>
            </ColumnFilters>
            </ig:Filtering>
        </Behaviors>
    </ig:Band>
</Bands>
```

Code Behind – Root Level

Execute in the `Page_Load` event handler

```
ColumnFilter filter = new
ColumnFilter(this.WebHierarchicalDataGrid1.GridView);
filter.ColumnKey = "ProductName";
filter.Condition = new RuleTextNode(TextFilterRules.Contains, "al");
this.WebHierarchicalDataGrid1.
         GridView.
         Behaviors.
         Filtering.
         ColumnFilters.
         Add(filter);
```

Code Behind – Band Level

Execute in the `InitializeBand` event handler

```
ColumnFilter filter = new ColumnFilter();
filter.ColumnKey = "CompanyName";
filter.Condition = new RuleTextNode(
       TextFilterRules.Contains,
       "al");
e.Band.Behaviors.Filtering.ColumnFilters.Add(filter);
```

Code Behind – Row Island Level

Execute in the `RowIslandCreated` event handler

```
ColumnFilter filter = new ColumnFilter();
filter.ColumnKey = "CompanyName";
filter.Condition = new RuleTextNode(
       TextFilterRules.Contains,
       "al");
e.RowIsland.Behaviors.Filtering.ColumnFilters.Add(filter);
```

	In JavaScript – Root Level
	Execute after the grid loads on the page.
	
	`var whdg = $find("WebHierarchicalDataGrid1");`
	`var topRowIsland = whdg.get_gridView();`
	`var filtering = topRowIsland.get_behaviors().get_filtering();`
	`var cf = filtering.create_columnFilter("ProductName")`
	`cf.get_condition().set_rule($IG.TextFilterRules.Contains);`
	`cf.get_condition().set_value("al");`
	`filtering.add_columnFilter(cf);`

Row Numbering

WebDataGrid	WebHierarchicalDataGrid
1. Enable Row Selectors 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Row Selection** 1.3.1. Set `RowNumbering` equal to `True`	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

Row Summaries

WebDataGrid	WebHierarchicalDataGrid
2. Enable Row Selectors 2.1. Open the **Smart Tag** 2.2. Select **Edit Behaviors** 2.3. Check **Summary Row**	← Same for parent grid To add behavior to child bands, see: *Adding Behaviors to WebHierarchicalDataGrid Child Bands*.

Selection: Cells

WebDataGrid	WebHierarchicalDataGrid
1. Enable the Selection Behavior 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Selection**	**1. Enable the Selection Behavior** 1.1. Open the **Smart Tag** 1.2. Select **Edit Behaviors** 1.3. Check **Selection**

Once selection is enabled on the grid then you may select cells on the server or client.

On the Server

Get Selected Cells

```
SelectedCellCollection selectedCells =
        this.WebDataGrid1.
                Behaviors.
                Selection.
                SelectedCells;
```

Select a Cell

```
selectedCells.Add(this.WebDataGrid1.Rows[0].Items[1]);
```

Deselect a Cell

```
selectedCells.RemoveAt(0);
```

On the Client

Get Selected Cells

```
var grid = $find('<%= WebDataGrid1.ClientID %>');

var selectedCells = grid.get_behaviors().
        get_selection().
        get_selectedCells();
```

Select a Cell

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var cell = grid.get_rows().get_row(0).get_cell(1);

grid.get_behaviors().
        get_selection().
        get_selectedCells().
        add(cell);
```

Deselect a Cell

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var cell = grid.get_behaviors().
                get_selection().
                get_selectedCells().
                getItem(0);

grid.get_behaviors().
        get_selection().
```

Once selection is enabled on the grid then you may select cells on the server or client.

On the Server

In order to have access to the columns for selection, the following snippets must be placed in the **RowIslandDataBound** event.

Get Selected Cells

```
Infragistics.Web.UI.GridControls.SelectedCellCollection selectedCells;

// Selecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selectedCells = e.RowIsland.Behaviors.Selection.SelectedCells;
}

// Selecting a column in the CHILD band
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selectedCells = e.RowIsland.Behaviors.Selection.SelectedCells;
}
```

Select a Cell

```
Infragistics.Web.UI.GridControls.Selection selection;

// Selecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedCells.Add(e.RowIsland.Rows[0].Items[0]);
}

// Selecting a column in the CHILD band
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedCells.Add(e.RowIsland.Rows[0].Items[0]);
}
```



```
get_selectedCells().
remove(cell);
```

On the Client

Get Selected Cells

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var childGrid = grid.get_gridView().
                        get_rows().
                        get_row(0).
                        get_rowIslands(0)[0];
var selectedCells;



// PARENT cell selection
if (parentGrid != null) {
        selectedCells = parentGrid.get_behaviors().
                                get_selection().
                                get_selectedCells();
}

// CHILD cell selection
if (childGrid != null){
        selectedCells = childGrid.get_behaviors().
                                get_selection().
                                get_selectedCells();
}
```

Select a Cell

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var childGrid = grid.get_gridView().
                        get_rows().
                        get_row(0).
                        get_rowIslands(0)[0];


// PARENT cell selection
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();

if (parentGrid != null) {
        var cell = parentGrid.get_rows().
                        get_row(0).
                        get_cell(0);

        parentGrid.get_behaviors().
                get_selection().
```

```
                    get_selectedCells().
                    add(cell);
}

// CHILD cell selection
if (childGrid != null) {
        var cell = childGrid.get_rows().
                            get_row(0).
                            get_cell(0);

        childGrid.get_behaviors().
                    get_selection().
                    get_selectedCells().
                    add(cell);
}
```

Selection: Columns

WebDataGrid

1. **Enable the Selection Behavior**
   1.1. Open the **Smart Tag**
   1.2. Select **Edit Behaviors**
   1.3. Check **Selection**

Once selection is enabled on the grid then you may select columns on the server or client.

On the Server

Get Selected Columns

```
SelectedColumnCollection selectedColumns =
        this.WebDataGrid1.
                Behaviors.
                Selection.
                SelectedColumns;
```

Select a Column

```
selectedColumns.Add(this.WebDataGrid1.Columns[0]);
```

Deselect a Column

```
selectedColumns.RemoveAt(0);
```

WebHierarchicalDataGrid

1. **Enable the Selection Behavior**
   1.1. Open the **Smart Tag**
   1.2. Select **Edit Behaviors**
   1.3. Check **Selection**

Once selection is enabled on the grid then you may select columns on the server or client.

On the Server

In order to have access to the columns for selection, the following snippets must be placed in the **RowIslandDataBound** event.

Get Selected Columns

```
Infragistics.Web.UI.GridControls.SelectedColumnCollection selectedCols;

// Selecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selectedCols = e.RowIsland.Behaviors.Selection.SelectedColumns;
}

// Selecting a column in the CHILD band
```

On the Client

Get Selected Columns

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var columns = grid.
                get_behaviors().
                get_selection().
                get_selectedColumns();
```

Select a Column

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var column = grid.get_columns().get_column(0);

grid.get_behaviors().
        get_selection().
        get_selectedColumns().
        add(column);
```

Deselect a Column

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var column = grid.
                get_behaviors().
                get_selection().
                get_selectedColumns().
                getItem(0);

grid.get_behaviors().
        get_selection().
        get_selectedColumns().
        remove(column);
```

```
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selectedCols = e.RowIsland.Behaviors.Selection.SelectedColumns;
}
```

Select a Column

```
Infragistics.Web.UI.GridControls.Selection selection;

// Selecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedColumns.Add(e.RowIsland.Columns[2]);
}

// Selecting a column in the CHILD band
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedColumns.Add(e.RowIsland.Columns[2]);
}
```

Deselect a Column

```
Infragistics.Web.UI.GridControls.Selection selection;

// Deselecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedColumns.Remove(e.RowIsland.Columns[2]);
}

// Deselecting a column in the CHILD band
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedColumns.Remove(e.RowIsland.Columns[2]);
}
```

On the Client

Get Selected Columns

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var childGrid = grid.get_gridView().
                              get_rows().
                              get_row(3).
                              get_rowIslands(0)[0];
var selectedColumns;


// PARENT column selection
if (parentGrid!= null) {
         selectedColumns = parentGrid.
                              get_behaviors().
                              get_selection().
                              get_selectedColumns();
}

// CHILD column selection
if (childGrid != null) {
         selectedColumns = childGrid.get_behaviors().
                  get_selection().
                  get_selectedColumns();
}
```

Select a Column

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var childGrid = grid.get_gridView().
                              get_rows().
                              get_row(3).
                              get_rowIslands(0)[0];
var column;

// PARENT column selection
if (parentGrid!= null){
         column = parentGrid.get_columns().get_column(0);
         parentGrid.
                  get_behaviors().
                  get_selection().
                  get_selectedColumns().
                  add(column);
}
```


	`// CHILD column selection`
	`if (childGrid != null) {`
	`  column = childGrid.get_columns().get_column(0);`
	`  childGrid.get_behaviors().`
	`  get_selection().`
	`  get_selectedColumns().`
	`  add(column);`
	`}`

Selection: Rows

WebDataGrid	WebHierarchicalDataGrid

1. Enable the Selection Behavior
1.1. Open the **Smart Tag**
1.2. Select **Edit Behaviors**
1.3. Check **Selection**
 	
Once selection is enabled on the grid then you may select rows on the server or client.
 	
On the Server
Get Selected Rows
`SelectedRowCollection selectedRows =`
`  this.WebDataGrid1.`
`  Behaviors.`
`  Selection.`
`  SelectedRows;`
 	
Select a Row
`selectedRows.Add(this.WebDataGrid1.Rows[0]);`
 	
Deselect a Row
`selectedRows.RemoveAt(0);`
 	
On the Client
Get Selected Rows
`var grid = $find('<%= WebDataGrid1.ClientID %>');`
`var selectedRows = grid.get_behaviors().`
`  get_selection().`
	1. Enable the Selection Behavior
	1.1. Open the **Smart Tag**
	1.2. Select **Edit Behaviors**
	1.3. Check **Selection**
	
	Once selection is enabled on the grid then you may select rows on the server or client.
	
	On the Server
	In order to have access to the columns for selection, the following snippets must be placed in the **RowIslandDataBound** event.
	
	Get Selected Rows
	`Infragistics.Web.UI.GridControls.SelectedRowCollection selectedRows;`
	
	`// Selecting a column in the PARENT band`
	`if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")`
	`{`
	`  selectedRows = e.RowIsland.Behaviors.Selection.SelectedRows;`
	`}`
	
	`// Selecting a column in the CHILD band`
	`if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"`
	`  && e.RowIsland.ParentRow ==`
	`this.WebHierarchicalDataGrid1.GridView.Rows[0])`
	`{`
	`  selectedRows = e.RowIsland.Behaviors.Selection.SelectedRows;`
	`}`


```
                        get_selectedRows();
```

Select a Row

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var row = grid.get_rows().get_row(0);

grid.get_behaviors().
        get_selection().
        get_selectedRows().
        add(row);
```

Deselect a Row

```
var grid = $find('<%= WebDataGrid1.ClientID %>');
var row = grid.
                get_behaviors().
                get_selection().
                get_selectedRows().
                getItem(0);

grid.get_behaviors().
        get_selection().
        get_selectedRows().
        remove(row);
```

Select a Row

```
Infragistics.Web.UI.GridControls.Selection selection;

// Selecting a column in the PARENT band
if (e.RowIsland.DataMember == "SqlDataSource1_DefaultView")
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedRows.Add(e.RowIsland.Rows[0].Items[0]);
}

// Selecting a column in the CHILD band
if (e.RowIsland.DataMember == "SqlDataSource2_DefaultView"
    && e.RowIsland.ParentRow ==
this.WebHierarchicalDataGrid1.GridView.Rows[0])
{
    selection = e.RowIsland.Behaviors.Selection;
    selection.SelectedRows.Add(e.RowIsland.Rows[0].Items[0]);
}
```

On the Client

Get Selected Rows

```
var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
var parentGrid = grid.get_gridView();
var childGrid = grid.get_gridView().
                        get_rows().
                        get_row(3).
                        get_rowIslands(0)[0];
var selectedRows;

// PARENT row selection
if (parentGrid!= null){
        selectedRows = parentGrid.
                        get_behaviors().
                        get_selection().
                        get_selectedRows();
}

// CHILD row selection
if (childGrid != null){
        selectedRows = childGrid.
                        get_behaviors().
                        get_selection().
```


	get_selectedRows();
	}
	
	Select a Row
	var grid = $find('<%= WebHierarchicalDataGrid1.ClientID %>');
	var parentGrid = grid.get_gridView();
	var childGrid = grid.get_gridView().
	get_rows().
	get_row(3).
	get_rowIslands(0)[0];
	var row;
	
	// PARENT row selection
	if (parentGrid!= null){
	row = parentGrid.get_rows().get_row(0);
	parentGrid.
	get_behaviors().
	get_selection().
	get_selectedRows().
	add(row);
	}
	
	// CHILD row selection
	if (childGrid != null){
	row = childGrid.get_rows().get_row(0);
	childGrid.
	get_behaviors().
	get_selection().
	get_selectedRows().
	add(row);
	}

Templating: Column Template

WebDataGrid	WebHierarchicalDataGrid
Column templates are possible via `TemplateDataField` columns or by assigning a template defined in the grids `Templates` collection to a particular column. **Using the TemplateDataField** **1. Create a TemplateDataField using the designer**	← Same

1.1. Open the **SmartTag**
1.2. Select **Edit Columns**
1.3. From the **Available Fields** pane, select **TemplateField**
1.4. Click **Add**
1.5. Set the `Key` to a unique value
1.6. Click **OK**

2. Switch to Code View to configure the template field

```
<Columns>
    <ig:TemplateDataField Key="Template1">
        <ItemTemplate>
            <asp:Button id="btnSelect" Text="Select" runat="server" />
            <%# DataBinder.Eval(
             ((Infragistics.Web.UI.TemplateContainer)Container).
                        DataItem,
                        "ProductName") %>
        </ItemTemplate>
    </ig:TemplateDataField>
</Columns>
```

Using Templates Collection

1. Create a new grid Template

1.1. Open the **SmartTag**
1.2. Select **Edit Template Collection**
1.3. Click **Add New Item**
1.4. Set the `TemplateID` to a unique value
1.5. Click **OK**

2. Switch to Code View and configure the template

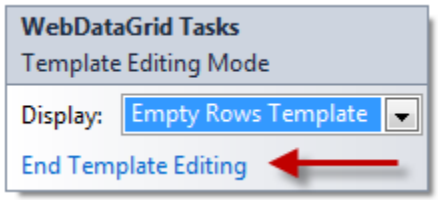
```
<ig:ItemTemplate ID="WebDataGrid1Template1" runat="server"
    TemplateID="formattedName">
        <Template>
            <div class="productName">
                <%# DataBinder.Eval(
                ((Infragistics.Web.UI.TemplateContainer)Container).
                                DataItem,
                                "ProductName") %>
            </div>
        </Template>
</ig:ItemTemplate>
```


3. Apply the Template in the InitializeRow event

```
private int NAME_COLUMN_INDEX = 0;

protected void WebDataGrid1_InitializeRow(object sender,
Infragistics.Web.UI.GridControls.RowEventArgs e)
{
    e.Row.Items[NAME_COLUMN_INDEX].TemplateId = "formattedName";
}
```

Note: Use either the `TemplateDataField` or a template from Templates collection on a single column as the `TemplateDataField` always takes precedence.

Templating: Empty Rows

WebDataGrid	WebHierarchicalDataGrid
1. Create the Empty Rows Template using the designer 1.1. Open the **Smart Tag** 1.2. Select **Edit Templates** 1.3. Select **Empty Rows Template** from the **Display** drop down 1.4. Configure the template in **Design** or **Source** view **Note:** When you are finished configuring the template in Design view click **End Template Editing** on the **Smart Tag** to return to the default Designer view of the grid.  Create the Empty Rows Template in code:	← Same


```
protected override void OnInit(EventArgs e)
{
    base.OnInit(e);
    //template needs to be instantiated on every postback
    this.WebDataGrid1.EmptyRowsTemplate = new
CustomEmptyRowsTemplate();
}

private class CustomEmptyRowsTemplate : ITemplate
{
    public void InstantiateIn(Control container)
    {
        System.Web.UI.WebControls.Label label1 = new
System.Web.UI.WebControls.Label();
        label1.Text = "Empty row template";
        label1.ID = "Label1";

        container.Controls.Add(label1);
    }
}
```

Grid Events

WebDataGrid	WebHierarchicalDataGrid	Events	Args
	X	ContainerGridDataBinding	DataBindingEventArgs
X		CustomDataBinding	DataBindingEventArgs
X	X	DataBinding	EventArgs
X	X	DataBound	EventArgs
X	X	Disposed	EventArgs
	X	GroupedColumnsChanged	GroupedColumnsChangedEventArgs
	X	GroupedColumnsChanging	GroupedColumnsChangingEventArgs
	X	GroupedRowInitialized	GroupedRowEventArgs
X	X	HeaderCheckBoxClicked	HeaderCheckBoxEventArgs
X	X	Init	EventArgs
	X	InitializeBand	BandEventArgs
X	X	InitializeRow	RowEventArgs
X	X	ItemCommand	HandleCommandEventArgs
X	X	Load	EventArgs
X	X	PreRender	EventArgs
	X	RowCollapsed	ContainerRowEventArgs
	X	RowExpanded	ContainerRowEventArgs
	X	RowIslandsCreated	RowIslandEventArgs
	X	RowIslandsDataBinding	RowIslandEventArgs
	X	RowIslandsDataBound	RowIslandEventArgs

	X	RowIslandsPopulated	ContainerRowEventArgs
	X	RowIslandsPopulating	ContainerRowCancelEventArgs
X	X	Unload	EventArgs

Events by Behavior

As the `WebHierarchicalDataGrid` extends the `WebDataGrid`, some events share event arg classes.

Behavior	Client Events (Args)	Arg Links	Server Events (Args)	Arg Links
Activation	ActiveCellChanged (ActiveCellChangedEventArgs)	WDG WHDG	ActiveCellChanged (ActiveCellEventArgs)	Same
	ActiveCellChanging (ActiveCellChangingEventArgs)	WDG WHDG	None	
	None		ActiveGroupedRowChanged (ActiveGroupedRowEventArgs)	Same
Cell Editing	EnteredEditMode (EditModeEventArgs)	WDG WHDG	*Uses Editing Core events*	
	EnteringEditMode (CancelEditModeEventArgs)	WDG WHDG		
	ExitedEditMode (EditModeEventArgs)	WDG WHDG		
	ExitingEditMode (CancelEditModeEventArgs)	WDG WHDG		
	Initialize	None		
Clipboard	Copied Copying Cut Cutting Initialize Pasted	None	None	

	Pasting			
Column Fixing	FixedStateChanged (FixedEvenArgs)	Same	FixedStateChanged (FixedStateChangedEventArgs)	Same
	FixedStateChanging (FixingEventArgs)	Same	None	
	Initialize	None		
Column Moving	HeaderDragEnd (HeaderDragEndEventArgs)	Same	None	
	HeaderDragStart (HeaderDragStart)	Same		
	HeaderDropped (HeaderDroppedEventArgs)	Same		
	HeaderMove (HeaderMoveEventArgs)	Same		
	Initialize	None		
	None		ColumnMoved (ColumnMovingEventArgs)	Same
Column Resizing	ColumnResized (ColumnResizedEventArgs)	Same	None	
	ColumnResizeDragging (ColumnResizeDraggingEventArgs)	Same		
	ColumnResizing (ColumnResizingEventArgs)	Same		
	Initialize	None	ColumnResized (ColumnResizingEventArgs)	Same
Editing Core	RowAdded (RowAddedEventArgs)	WDG WHDG	RowAdded (RowAddedEventArgs)	WDG WHDG
	RowAdding	None	RowAdding (RowAddingEventArgs)	Same
	RowDeleted	None	RowDeleted (RowDeletedEventArgs)	Same
	RowDeleting	None	RowDeleting (RowDeletingEventsArgs)	Same
	RowUpdated (RowUpdatedEventArgs)	WDG WHDG	RowUpdated (RowUpdatedEventArgs)	Same
	RowUpdating	None	RowUpdating (RowUpdatingEventsArgs)	WDG WHDG
	CellValueChanged (CellValueChangedEventArgs)	WDG WHDG	None	

	CellValueChanging (CancelCellValueChangingEventArgs)	WDG WHDG		
	Initialize	None		
Filtering	DataFiltered (DataFilteredArgs)	Same	DataFiltered (FilteredEventArgs)	Same
	DataFiltering (CancelApplyFiltersEventArgs)	Same	DataFiltering (FilteringEventArgs)	Same
	ColumnFilterAdded (ColumnFilterAddedArgs)	Same	None	
	EnteredEditMode (EditModeEventArgs)	WDG WHDG		
	EnteringEditMode (CancelEditModeEventArgs)	WDG WHDG		
	ExitedEditMode (EditModeEventArgs)	WDG WHDG		
	ExitingEditMode (CancelEditModeEventArgs)	WDG WHDG		
	FilterDropdownDisplayed FilterDropdownDisplaying FilterDropDownHidden FilterDropDownHiding Initialize	None		
Paging	Initialize	None	None	
	PageIndexChanging (PagerEventArgs)	WDG WHDG		
	PageIndexChanged	None	PageIndexChanged (PagingEventArgs)	Same
Row Adding	EnteredEditMode (EditModeEventArgs)	WDG WHDG	None	
	EnteringEditMode (CancelEditModeEventArgs)	WDG WHDG		
	ExitedEditMode (EditModeEventArgs)	WDG WHDG		

	ExitingEditMode (CancelEditModeEventArgs)	WDG WHDG		
	Initialize	None	*Uses Editing Core events*	
Row Deleting	None		*Uses Editing Core events*	
Row Editing Template	Initialize	None	None	
	TemplateClosed (EditRowEventArgs)	WDG WHDG		
	TemplateClosing (CancelEditRowEventArgs)	WDG WHDG		
	TemplateOpened (EditRowEventArgs)	WDG WHDG		
	TemplateOpening (CancelEditRowEventArgs)	WDG WHDG	*Uses Editing Core events*	
Row Selectors	FooterRowSelectorClicked (MarginRowSelectorClickedEventArgs)	Same	None	
	HeaderRowSelectorClicked (MarginRowSelectorClickedEventArgs)	Same		
	Initialize	None		
	RowSelectorClicked (RowSelectorClickedEventArgs)	Same		
	RowSelectorClicking (RowSelectorClickingEventArgs)	Same		
Selection	CellSelectionChanging (CellSelectionChangingEventArgs)	WDG WHDG	None	
	ColumnSelectionChanging (ColumnSelectionChangingEventArgs)	WDG WHDG		
	Initialize	None		
	RowSelectionChanging (RowSelectionChangingEventArgs)	WDG WHDG		
	CellSelectionChanged (CellSelectionChangedEventArgs)	WDG WHDG	CellSelectionChanged (SelectedCellEventArgs)	Same

	ColumnSelectionChanged (ColumnSelectionChangedEventArgs)	WDG WHDG	ColumnSelectionChanged (SelectedColumnEventArgs)	Same
	RowSelectionChanged (RowSelectionChangedEventArgs)	WDG WHDG	RowSelectionChanged (SelectedRowEventArgs)	Same
Sorting	ColumnSorting (SortingEventArgs)	WDG WHDG	None	
	Initialize	None		
	ColumnSorted (SortedEventArgs)	WDG WHDG	ColumnSorted (SortingEventArgs)	WDG WHDG
Summary Row	Initialize SummaryCalculated SummaryDropDownDisplayed SummaryDropDownDisplaying SummaryDropDownHidden SummaryDropDownHiding	None	CalculateSummary (CustomSummaryEventArgs)	Same
			SummaryRowCalculated (SummaryEventArgs)	Same
	CalculateCustomSummary		CalculateCustomSummary (CustomSummaryEventArgs)	Same
Virtual Scrolling	FormatToolTip (VirtualScrollingFormatToolTipEventArgs)	Same	None	
	Initialize	None		
	MoreRowsReceived (MoreRowsRecievedEventArgs)	Same		
	MoreRowsRequesting (CancelMoreRowsRequestingEventArgs)	Same		

Revision History

- 07/08/2011:
  - Added snippet sections:
    - Ajax: Load on Demand (Manual)
    - Editing: Adding Rows
    - Column: Summaries
  - Added tables:
    - Grid Events
    - Events by Behavior

