

WebGrid.NET Enterprise 6.0 (2008 R2 Edition)

As the flagship product of Intersoft Solutions Corp., WebGrid.NET Enterprise is continuously enhanced with new features and enhancements to support more scenarios specifically in enterprise and complex Web application development.

The following sections discuss the enhancements made available to WebGrid.NET Enterprise in 2008 R2 release cycle.

Table of Contents

Strong support for Accessibility and Compliance with Section 508 Standards.	2
Enabling Section 508 Standards in WebGrid	3
Using new properties for assistive technology requirement.....	3
Automatic Group Total Update.....	5
Preventing new row with default values to be marked as edit	5
WebCombo.NET as FilterEditType	6
Multiple Row Selection	6
How-to: Access the selected rows in client side and server side	8
Other enhancements related to Multiple Row Selection feature	9
Preserving checked/selected rows on FlyPostBack™ actions.....	10
New “ButtonImage” Column Type.....	11
How-To: Use ButtonImage column type and assign the Image of the Button.	11
Option to turn off animation in context menu	12
Preserving expanded child rows	12

Strong support for Accessibility and Compliance with Section 508 Standards.

WebGrid.NET Enterprise now fully implements the support for Section 508 Standards in the following provision:

- Provision 1194.22 – About Data Table Headers & Associations
 - Row and column headers should be identified for data tables.
 - Column headers should be in the same table structure as the table's contents (rows).
 - Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.
- Objectives:
 - Allows Web developers with their assistive technology to build solution using WebGrid.NET Enterprise for supporting people with visual disabilities.
 - Allows users of screen readers to interpret table data efficiently.

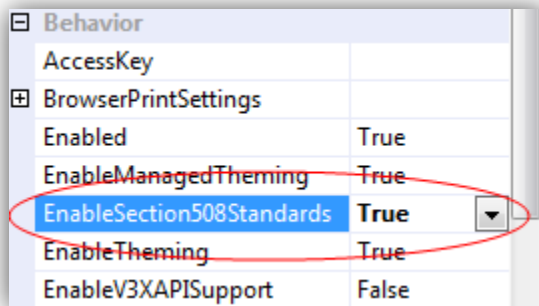
Development notes and changes in WebGrid when this new feature is enabled:

- WebGrid renders column headers inside the content's table structure. The header and content structure is no longer separated.
- The fixed column header is no longer applicable. This means that when you scroll downward to the bottom, the column headers will follow as well.
- Column headers appearance and styles will still be rendered in the same fashion as in normal mode.
- Other type or headers such as filter row and new row is now rendered inside the content's table structure as well.
- Basic column operations are still supported in this accessibility mode, such as column moving, resizing, sorting, grouping and filtering.
- Working perfectly in conjunction with the following advanced features:
 - User Interactions using keyboard, context menu, mouse selection, and drag drop.
 - Virtual Load.
 - Column Freezing.
 - AutoFit Columns.
 - All FlyPostBack actions such as refresh, grouping, load more data, etc.
- Hierarchical and Self Referencing modes are supported.
- Client side API is internally restructured, no public API changes are necessary. This means, your existing codes should run properly without any changes required.
- ColumnSet layout mode is not supported in this version.

Enabling Section 508 Standards in WebGrid

WebGrid.NET Enterprise makes it very easy for developers to enable Section 508 Standards mode in WebGrid instances. A new property at the WebGrid level has been added to take advantage of this mode.

To enable Section 508 Standards mode in your WebGrid, simply locate the *EnableSection508Standards* property in Visual Studio property window, and set it to *True*.



When you run the WebForm in the browser, you wouldn't notice significant difference with the normal mode, except the column headers are now scrolled when the vertical scrollbar of the Grid is scrolled downward.

While having significant changes in WebGrid's internal structures and APIs to support this new Accessibility mode, WebGrid is still fully compatible with all existing features such as column moving, grouping, resizing, filtering, sorting, context menu and the rest of advanced features.

Thanks to the rock solid architecture of WebGrid, it allows all existing public APIs to function properly without requiring users to modify their existing codes.

Using new properties for assistive technology requirement

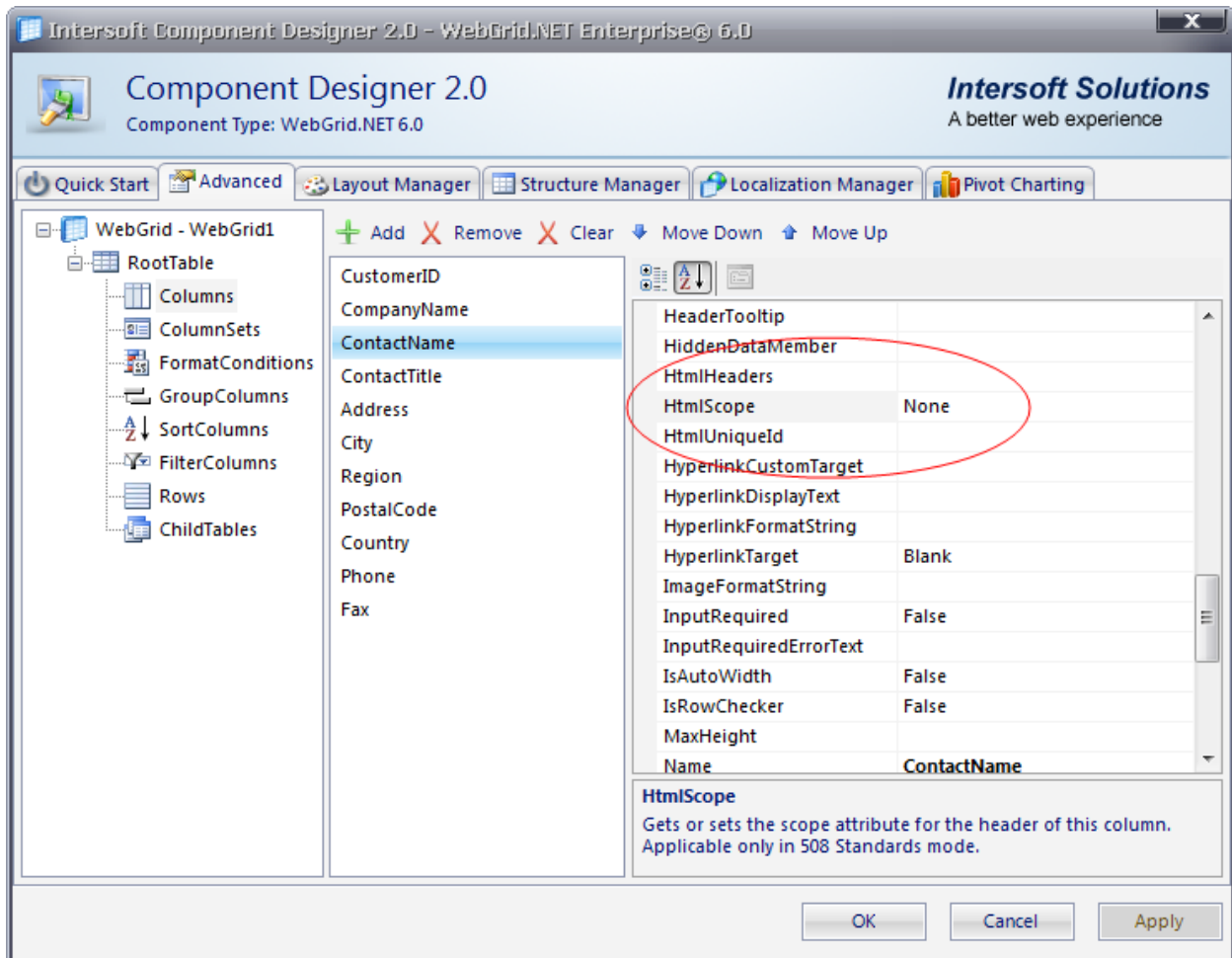
WebGrid.NET Enterprise 6.0 also exposes several new properties at the Column level to provide flexibility and customization for developers to build their own assistive technology, which connected into the WebGrid instance.

The new properties for *WebGridColumn* are listed in the following:

- *HtmlScope*. When specified, WebGrid will render the *scope* attribute to the column header. Valid values are *None*, *Row*, *Col*, *RowGroup* and *ColGroup*.
- *HtmlUniqueId*. When specified, WebGrid will render the *id* attribute to the column header according to the value of this property.
- *HtmlHeaders*. When specified, WebGrid will render the *headers* attribute to the column header according to the value of this property.

To access the new properties of WebGridColumn in designtime, please do the following:

- Right click on the WebGrid instance.
- Choose WebGrid.NET Designer.
- Click on Advanced tab.
- Expand WebGrid, RootTable, and choose Columns.
- Select a column.
- Click “See all properties” in the right panel containing properties. Please refer to the following screenshot for more details.

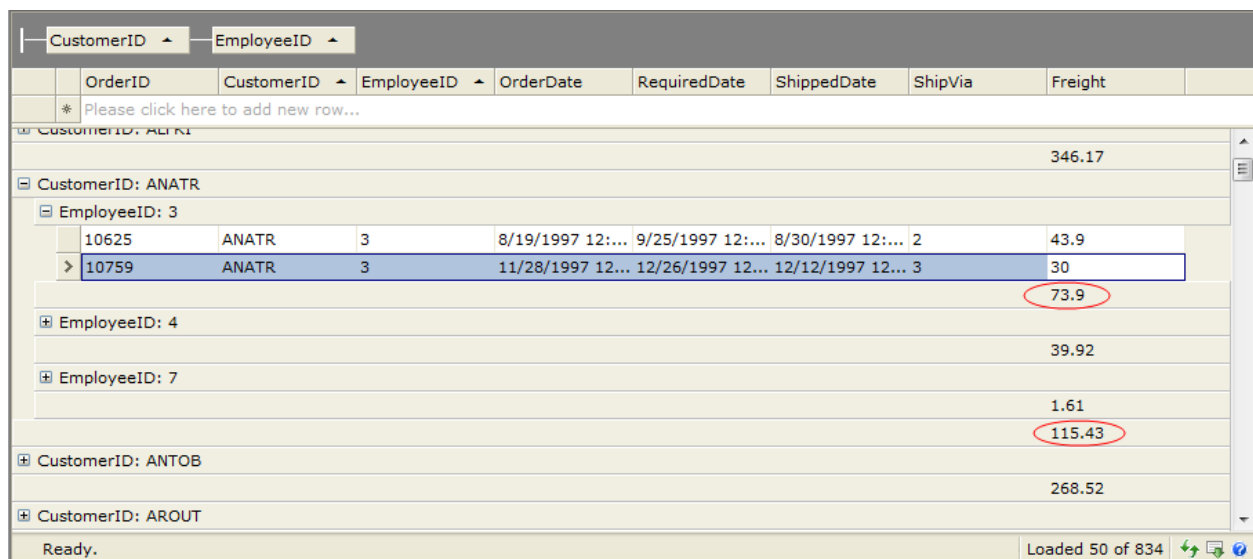


Automatic Group Total Update

WebGrid.NET Enterprise is the first ASP.NET DataGrid that introduces rich data display features, such as grouping with built-in aggregation. In addition, WebGrid also allows developers to show GroupTotal in the data display. When enabled, every group will show a total row at the end of its group. This allows users to easily review the aggregate results of each column, per that group.

New to this release is the ability to automatically update the group total and display the aggregate results correctly after data editing. This new enhancement removes the workaround needed to calculate the group total.

This new feature also works in multiple grouping mode. That means, if you have 3 group columns, WebGrid will still be able to update the aggregate results of all affected group totals.



The screenshot displays a WebGrid with two grouped columns: CustomerID and EmployeeID. The grid shows a hierarchy of data with group totals. A red circle highlights the Freight value 30 in the row for EmployeeID 3, CustomerID ANATR. Another red circle highlights the updated group total 115.43 for CustomerID ANATR. The status bar at the bottom indicates 'Loaded 50 of 834'.

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight
CustomerID: ANATR							346.17
EmployeeID: 3							
10625	ANATR	3	8/19/1997 12:...	9/25/1997 12:...	8/30/1997 12:...	2	43.9
10759	ANATR	3	11/28/1997 12:...	12/26/1997 12:...	12/12/1997 12:...	3	30
EmployeeID: 4							73.9
EmployeeID: 7							39.92
CustomerID: ANTOB							1.61
CustomerID: AROUT							115.43
CustomerID: ANTOB							268.52
CustomerID: AROUT							

The above screenshot shows a Grid with two grouped columns. When changes is made to the Freight column, the group total for both EmployeeID and CustomerID (as indicated in the red circle) are automatically updated.

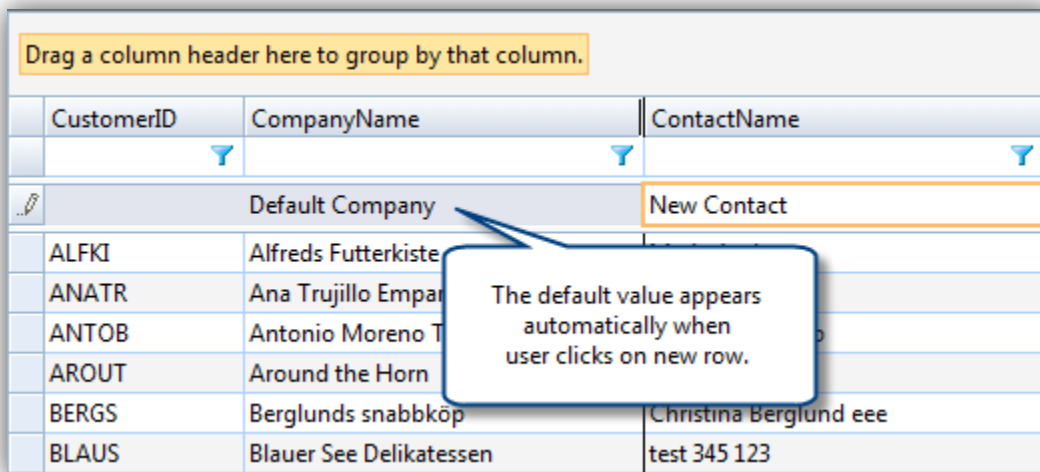
Preventing new row with default values to be marked as edit

WebGrid has a built-in feature to allows developers to specify the default value that displayed when user type in the NewRow area. This reduces data entry effort when the default value matches the most of the new values to input.

A grid with default value in one of the columns is causing issue when the user set the cursor to the NewRow in the top of the grid and removes it by clicking somewhere else in the grid. In the current state the NewRow would be saved even if the user doesn't entered any data, especially when *NewRowLostFocusAction* property is set to *AlwaysUpdate*.

WebGrid now includes a new property *MarkDefaultValuesAsDirty* property inside *LayoutSettings* to provide more flexibility for developers to control this specific behavior. This new property is set to true by default to preserve the compatibility with existing behavior.

To prevent WebGrid to mark the new row with default values as dirty (edited), simply set the *MarkDefaultValuesAsDirty* property to *false*.



Drag a column header here to group by that column.		
CustomerID	CompanyName	ContactName
	Default Company	New Contact
ALFKI	Alfreds Futterkiste	
ANATR	Ana Trujillo Empa	
ANTOB	Antonio Moreno T	
AROUT	Around the Horn	
BERGS	Berglunds snabbköp	Christina Berglund eee
BLAUS	Blauer See Delikatessen	test 345 123

WebCombo.NET as FilterEditType

WebGrid provides strong integration with other Intersoft products, such as ability to use WebCombo.NET as the editing type in the WebGrid. In addition, these integration features are applicable to filter editing type as well.

New in this version is the ability to specify *WebCombo.NET* as the *FilterEditType* individually. Previously you can only use WebCombo.NET as FilterEditType when the EditType is using WebCombo.NET. This introduces limitation when *EditType* is set to *NoEdit*.

With this enhancement in this new version, you can now set *EditType* to any editing type, and still be able to set *FilterEditType* to *WebCombo.NET* specifically.

Multiple Row Selection

WebGrid.NET provides a feature that allows users to select multiple rows using checkbox. With this initial approach, checkbox will appear in every row in the Grid to allow users to check on the checkbox in order to select the row.

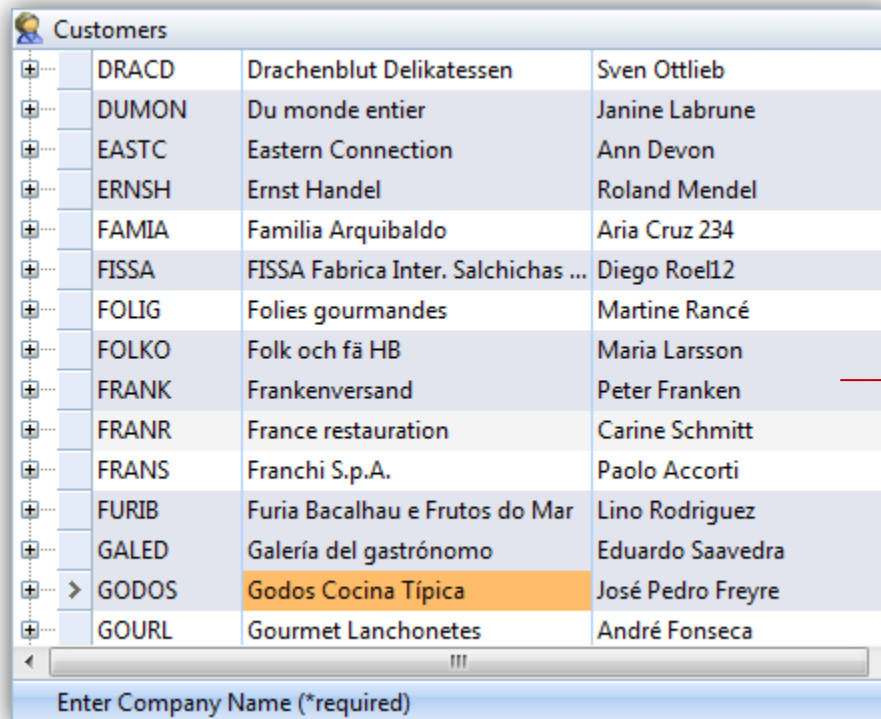
In several scenarios, developers often want the ability to select multiple rows without using checkbox due to application design. WebGrid.NET Enterprise™ in 2008 R2 release now allows you to perform multiple row selection by using the user-friendly Ctrl and Shift key combination. This feature allows you

to simulate Windows-like interactivity and elegant design to select multiple rows without the need to use checkbox.

IMPORTANT: Multiple Row Selection can't be used alongside with RowChecker feature, as both features will cause conflict and undesired effect to the selection result. You should enable only *RowChecker* feature (which used checkbox) or the new *Multiple Row Selection* feature at a time.

Keyboard combination supported in this new feature:

- Use Ctrl + Click to select any individual rows.



+	DRACD	Drachenblut Delikatessen	Sven Ottlieb
+	DUMON	Du monde entier	Janine Labrune
+	EASTC	Eastern Connection	Ann Devon
+	ERNSH	Ernst Handel	Roland Mendel
+	FAMIA	Familia Arquibaldo	Aria Cruz 234
+	FISSA	FISSA Fabrica Inter. Salchichas ...	Diego Roel12
+	FOLIG	Folies gourmandes	Martine Rancé
+	FOLKO	Folk och få HB	Maria Larsson
+	FRANK	Frankenversand	Peter Franken
+	FRANR	France restauration	Carine Schmitt
+	FRANS	Franchi S.p.A.	Paolo Accorti
+	FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez
+	GALED	Galería del gastrónomo	Eduardo Saavedra
+	➤ GODOS	Godos Cocina Típica	José Pedro Freyre
+	GOURL	Gourmet Lanchonetes	André Fonseca

Enter Company Name (*required)

To select individual rows, hold down Ctrl key, and then click on the row to select it.

- Use Shift + Click to select a range of continuous rows.

	Customer ID	Customer Name
+	DRACD	Drachenblut Delikatessen
+	DUMON	Du monde entier
+	EASTC	Eastern Connection
+	ERNSH	Ernst Handel
+	FAMIA	Familia Arquibaldo
+	FISSA	FISSA Fabrica Inter. Salchichas ...

You can easily select a range of continuous rows by clicking on the start row, then hold down Shift key, and click on the destination ending row.

- Use Ctrl + Shift + Click to select a range of continuous rows without clearing previous selection.

	Customer ID	Customer Name
+	DRACD	Drachenblut Delikatessen
+	DUMON	Du monde entier
+	EASTC	Eastern Connection
+	ERNSH	Ernst Handel
+	FAMIA	Familia Arquibaldo
+	FISSA	FISSA Fabrica Inter. Salchichas ...
+	FOLIG	Folies gourmandes
+	FOLKO	Folk och fä HB
+	FRANK	Frankenversand
+	FRANR	France restauration
+	FRANS	Franchi S.p.A.
+	FURIB	Furia Bacalhau e Frutos do Mar

Use the combination of Ctrl and Shift key to select a range of continuous rows while preserving the previous selection.

Thanks to the innovative architecture in WebGrid – you can enable this feature by simply setting the *AllowMultipleSelection* property to Yes value. This means you no longer require any workarounds or custom codes to enable this interactivity feature.

How-to: Access the selected rows in client side and server side

To get the selected rows in client side, you can use *GetCheckedRows()* method of a *WebGridTable* instance. It is the same method used in initial *RowChecker* feature.

Similarly, to get the selected rows in server side, you use the *GetCheckedRows()* method of a *WebGridTable* instance. See following example:

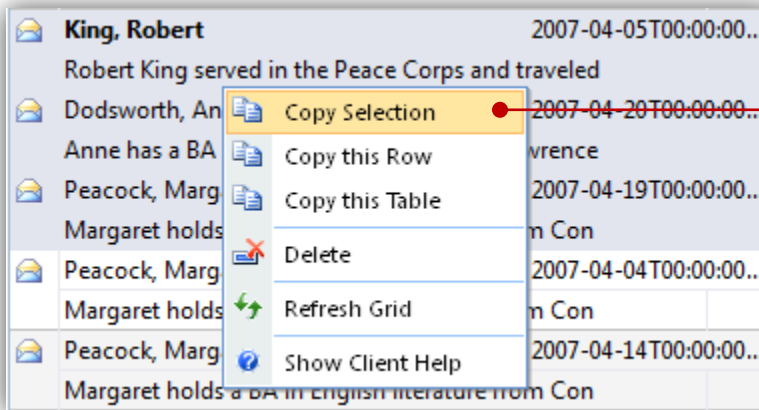
```
protected void button1_Click(object sender, EventArgs e)
{
    Response.Write(WebGrid1.RootTable.GetCheckedRows().Count);
}
```


Other enhancements related to Multiple Row Selection feature

The new Multiple Row Selection feature can be used in conjunction with many existing features. The bare-metal architecture of WebGrid allows a new feature to work flawlessly with other features, providing a great flexibility and options for developers to customize WebGrid's behaviors.

When Multiple Row Selection is enabled, the row context menu will show a new command item to allow users to copy the selected rows to the clipboard.

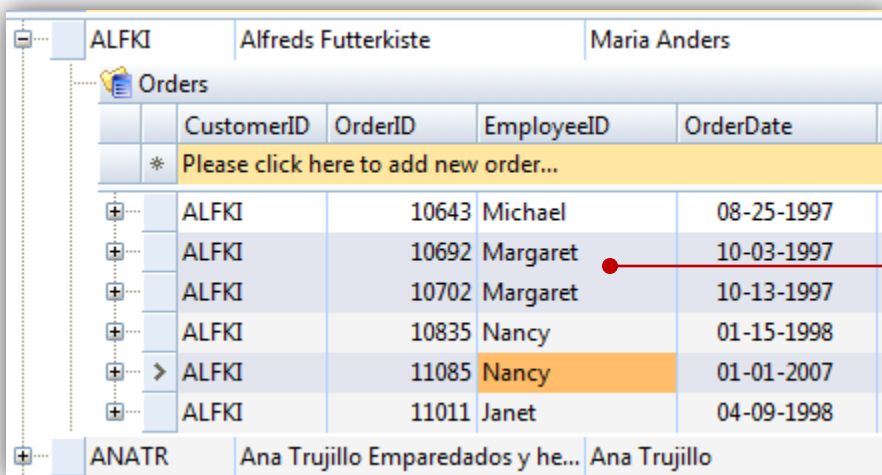
The following screenshot shows the *Multiple Selection* feature in a Grid with *ColumnSet* configuration.



The new "Copy Selection" menu item will appear when you right click on the selection (any selected rows area).

The Copy Selection function is a handy way to copy selected information to other sources. For instance, you can copy the selection to clipboard, and paste them to Excel worksheet or Word document.

By default, when *AllowMultipleSelection* property is set to *Yes*, all tables (*WebGridTable*) in the hierarchical configuration will apply the setting as well, thus allowing multiple selection on all tables. You can also disable the feature on individual table by setting the *AllowMultipleSelection* property of the desired table to *No* value.



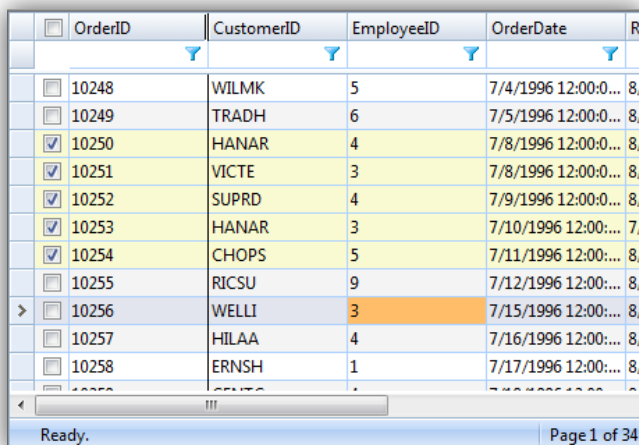
*The **Multiple Row Selection** feature can be enabled only in Child Table, while disabled in Root Table.*

Preserving checked/selected rows on FlyPostBack™ actions

Now that WebGrid has two ways to allow users to perform multiple row selection. The first is using *IsRowChecker* feature which is introduced back in WebGrid version 4.0. The latter one is using [Multiple Row Selection](#), which is a new feature in this release.

One important requirement related to the multiple row selection is the ability to persist the checked/selected rows between *postbacks* and *AJAX callbacks* (FlyPostBack). This requirement is especially useful when used in conjunction with *Classic Paging* feature. Users will then be able to check some rows in a page, moving to another page or perform sorting and check some more rows, and finally submit all checked/selected rows to server. The previous version of WebGrid already includes a function called *PersistRowChecker* to persist checked rows on postback. It is, however, applies only to Full Postback.

New to WebGrid.NET Enterprise 6.0 in this second 2008 release is the new feature to automatically preserve checked/selected rows on both *FullPostBack* and *FlyPostBack*. With this feature, users can now preserve the previously checked rows while performing other actions such as moving to other pages, grouping, sorting, and filtering and so on. Furthermore, WebGrid will mark back the checked rows when it found the previously checked rows in the page. It will also restore the checked selection so that users can easily notice on checked rows.



<input type="checkbox"/>	OrderID	CustomerID	EmployeeID	OrderDate	Row
<input type="checkbox"/>	10248	WILMK	5	7/4/1996 12:00:0...	8/
<input type="checkbox"/>	10249	TRADH	6	7/5/1996 12:00:0...	8/
<input checked="" type="checkbox"/>	10250	HANAR	4	7/8/1996 12:00:0...	8/
<input checked="" type="checkbox"/>	10251	VICTE	3	7/8/1996 12:00:0...	8/
<input checked="" type="checkbox"/>	10252	SUPRD	4	7/9/1996 12:00:0...	8/
<input checked="" type="checkbox"/>	10253	HANAR	3	7/10/1996 12:00:...	7/
<input checked="" type="checkbox"/>	10254	CHOPS	5	7/11/1996 12:00:...	8/
<input type="checkbox"/>	10255	RICSU	9	7/12/1996 12:00:...	8/
<input type="checkbox"/>	10256	WELLI	3	7/15/1996 12:00:...	8/
<input type="checkbox"/>	10257	HILAA	4	7/16/1996 12:00:...	8/
<input type="checkbox"/>	10258	ERNSH	1	7/17/1996 12:00:...	8/

The **checked rows** are now automatically restored after FlyPostBack™ actions.

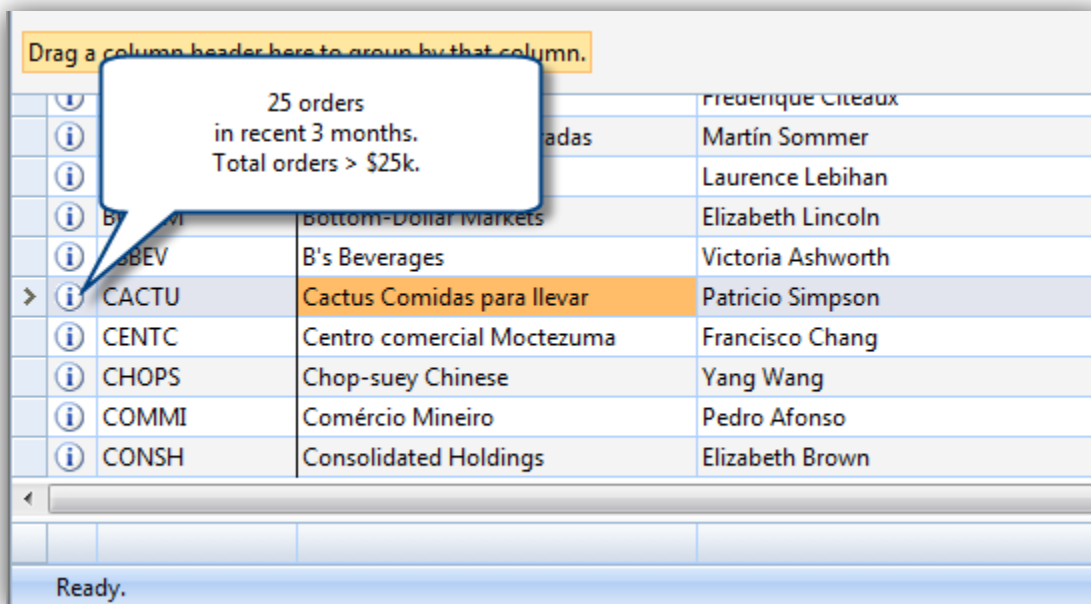
The setting to enable this feature is controlled by *RestoreRowSelection* property inside the *LayoutSettings* property of WebGrid. This property has 4 possible values:

- *Disabled*. The default value, which turns off the row selection restoration.
- *RootTableOnly*. Only checked rows in root table will be restored.
- *ChildTableOnly*. Only checked rows in child table will be restored.
- *All*. All checked rows in any table types will be restored.

New “ButtonImage” Column Type

Due to popular demand, WebGrid now includes a new *ButtonImage* column type for handy button display with image. Although developers can use *Template* column type to specify any kind of contents such as `<asp:Image>`, it doesn't take advantage of WebGrid's postback mechanism and developers have to handle the events manually, which is time consuming.

The following screenshot shows a WebGrid using ButtonImage column type in the first column.



The *ButtonImage* column type takes advantage of existing architecture such as *ButtonAutoPostBack* property and *ButtonPostBackMode* property. The *ButtonClick* client side event is invoked as well for *ButtonImage* column type.

The sample *ButtonImage* definition in *WebGridColumn* is as following:

```
<ISWebGrid:WebGridColumn Width="25px" ColumnType="ButtonImage"
    ButtonImage="images/info.gif" ButtonAutoPostBack="true">
</ISWebGrid:WebGridColumn>
```

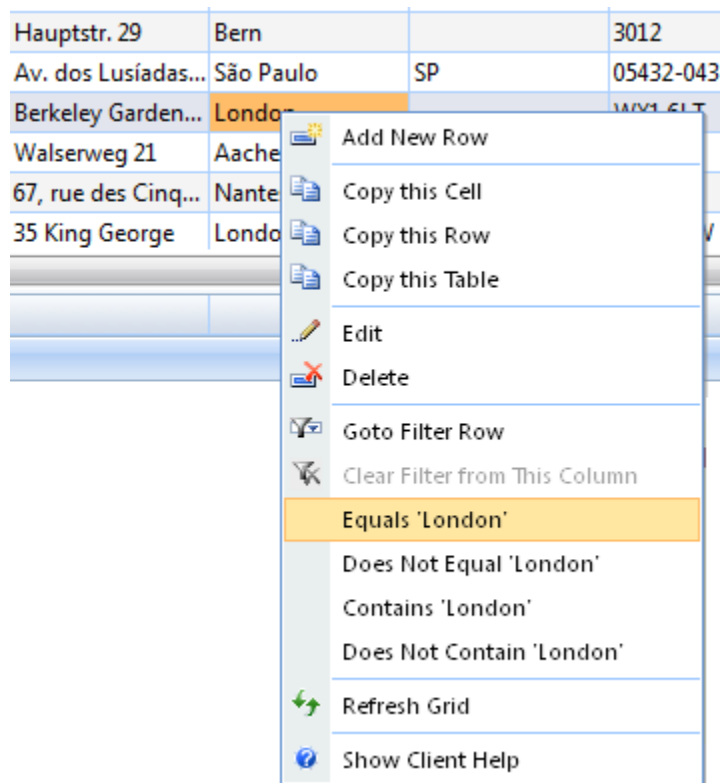
How-To: Use ButtonImage column type and assign the Image of the Button.

[TODO]

Option to turn off animation in context menu

One of the best features in WebGrid is its rich user interaction. WebGrid includes rich context menu natively, so that users can choose from various commands that related to a specific object or context. By default, the context menu will be shown with fading animation for best user experience.

The following image shows the context menu on row object with *AutoFilterSuggestion* feature enabled.



New to this version is the option to turn off animation in context menu. WebGrid now provides a property called *ContextMenuAnimation* inside the *LayoutSettings* property. You can set it to *false* value to turn off the animation in all WebGrid's context menu.

Preserving expanded child rows

WebGrid.NET Enterprise™ 6.0 in this new release includes a nice enhancement to preserve expanded child rows automatically. A new property named *RestoreExpandedChildRows* is now added to the *LayoutSettings*.

When enabled, WebGrid automatically tracks the child rows that end users expanded during user interaction. Next, when users performed any FlyPostBack™ actions – such as sorting, refresh, filtering, etc – WebGrid will restore any matching child rows that expanded previously. This long-awaited enhancement is also applicable to any level of nested tables.

To enable this feature, simply set *RestoreExpandedChildRows* property to *true* value.