WebGrid.NET Enterprise 5.0 White Paper

This whitepaper discusses breaking changes, enhancements and new features available in WebGrid.NET Enterprise 5.0.

Table of Contents

TABLE OF CONTENTS 1

I. BREAKING CHANGES IN WEBGRID.NET ENTERPRISE 5.0 3

II. UPGRADING FROM PREVIOUS VERSION 3

OVERVIEW 3

ASSEMBLY REFERENCES 4

III. WHAT'S NEW IN 5.0 4

NO-CODES DATA BINDING THROUGH DATASOURCECONTROL SUPPORT 4

Basic DataSourceControl support 5

Advanced DataSourceControl support 5

About ISDataSource licensing in WebGrid.NET Enterprise 5.0 7

The benefits of the new datasource control binding support 7

NO-CODES HIERARCHICAL DATA BINDING 9

Load all child tables data source (default) 9

Load child tables data source on demand 10

NO-CODES WEBVALUELIST DATA BINDING 13

DESIGN-TIME SUPPORT FOR DATASOURCECONTROL 13

SmartTag Action Panel 13

Retrieve Structure and Hierarchical Structure Context Command 14

Flat table datasource control 15

Xml datasource control 16

Hierarchical datasource control 18

NEW USER INTERFACE, NEW STYLES 19

COLUMN FREEZING 20

Main Concept 21

Runtime freezing/unfreezing 22

Absolute Scrolling 23

LiveFreeze™ behaviors and other notes 23

CLEANER AND REDUCED PAGE OUTPUT THROUGH BUILT-IN DEFAULT STYLE 25

Default Style Mode 26

FAQ: What does Default Style look like? *27*

How-to: Enable Default Style for existing instance of WebGrid *27*

HASSLE-FREE DEPLOYMENT THROUGH SMARTWEBRESOURCES™ TECHNOLOGY 28

Overview 28

About Localization Files 28
FAQ: Does SmartWebResources feature automatically enabled in WebGrid.NET Enterprise 5.0? 29
FAQ: Briefly, what are the files included in WebGrid SmartWebResources Assembly? 29
FAQ: I got a warning message "ContextMenu is disabled due to missing resources" after page load in the browser. What should I do to resolve this problem? 29
FAQ: I am aware that WebGrid.NET Enterprise 5.0 requiring WebDesktop SmartWebResources as part of its User Interface engine. Do I need to purchase a separate WebDesktop's license in order to use its SmartWebResources? 29
How-to: Configure SmartWebResources in a new web application 30
How-to: Disable SmartWebResources for Localization's language files 30
VERBOSE EDITING INFOTEXT 31
AUTOMATIC FILTER SUGGESTION 31
FILTER STATUS INDICATOR 32
UNBOUND HIERARCHICAL 32
LOADING PREVIEWROW ON DEMAND 32
DISPLAYING AGGREGATE VALUES IN GROUPHEADER 33
MULTIPLE VALUES SUPPORT IN WEBVALUELIST 33
MULTIPLE VALUES EDITING USING WEBCOMBO.NET 4.0 CONTROL 34
FAQ: Does WebCombo.NET 4.0 control included in WebGrid.NET Enterprise 5.0? 34
AUTOMATIC COLUMN SORTING USING WEBVALUELIST 34
BUILT-IN CELL VALIDATION THROUGH NEW REQUIRED INPUT 35
NEW EDIT TYPE "RESIZABLE TEXTBOX" 36
IMPROVED DATA CACHING (FOR TRADITIONAL DATA BINDING) 37
TEMPLATED CELL 37
WEBGRID.NET COMPONENT DESIGNER 2.0 39
Quick Start 40
Advanced 41
Layout Manager 43
Structure Manager 45
Localization Manager 46
ENHANCEMENTS 48
Server-side object model 48
Client-side object model 49
Client events and user interaction 49
Exporting 49
User Interface 50
Core engine on output optimization 50
Hierarchical 50
Editing 50
Filtering 51
Context Menu 51
Self Reference 51

I. Breaking changes in WebGrid.NET Enterprise 5.0

There are no breaking changes in WebGrid.NET Enterprise 5.0. All existing object model, features and behaviors remain the same.

II. Upgrading from previous version

Overview

WebGrid.NET Enterprise version 5.0 is the most significant major upgrade since initial version of WebGrid.NET containing key enhancements in user interface, enhanced grid functions, Excel® style column freezing, declarative data binding support and rich designer experience.

If you are planning to upgrade your existing web application to use WebGrid.NET Enterprise 5.0, please take the following points into consideration:

- WebGrid.NET Enterprise 5.0 is especially designed for ASP.NET 2.0 in order to fully support native databinding features and new design-time features to improve developer's productivity.
  If your existing web application is still running on ASP.NET 1.x (developed using Visual Studio 2003), you cannot upgrade to version 5.0. We strongly encourage developers to start migrating to Visual Studio 2005 to obtain many benefits and features introduced in .NET 2.0.

  If you are a new web developer and would like to develop ASP.NET 2.0 web application using WebGrid.NET Enterprise 5.0, you may want to check out Microsoft's Visual Web Developer Express. Visual Web Developer Express is a lightweight version of Visual Studio 2005, and is provided for free by Microsoft. Since that WebGrid.NET Enterprise 5.0 is built on the top of Visual Studio 2005 architecture, Visual Web Developer Express is fully supported as well. To download Visual Web Developer Express, visit http://msdn.microsoft.com/vstudio/express/vwd/download/

- Use new optimization features available in version 5.0 for better performance. Consider to enable the following features after your migration:
  - DefaultStyle. Enable DefaultStyle feature in WebGrid will significantly reduce page output size, especially when large number of WebGrid is placed in a webform.
  - SmartWebResources. When enabled, this feature will allow you to easily deploy WebGrid without the need of physical client scripts or client resources. With this feature, you can now develop ASP.NET 2.0 File System's website without additional efforts.

Assembly References

Our development team is working hard to ensure that you can upgrade your web application as smooth as possible in less efforts. You can easily migrate your existing web application to use version 5.0 by simply changing the assembly references to 5.0 assemblies.

To upgrade an existing ASP.NET 2.0 web application to use WebGrid.NET Enterprise 5.0, please do the following:

- Open an existing web application in Visual Studio 2005/Express
- Expand Bin folder
- Remove ISNet.dll, ISNet.WebUI.dll and ISNet.WebUI.WebGrid.dll
- Right click on the Project node then click Add Reference
- Add ISNet.dll, ISNet.WebUI.dll from [Installation Folder]\WebUI.NET Framework 3.0\Bin [Note that the file version should be 3.0.5000.400+]
- Add ISNet.WebUI.WebGrid.dll from [Installation Folder]\WebGrid.NET 5.0\Bin
- Rebuild your web application and run in browser

If you are planning to use new SmartWebResources™ feature available in WebGrid.NET Enterprise 5.0 for simpler deployment, please refer to Hassle-free deployment through SmartWebResources™ technology

If your existing web application used WebGrid's Exporting feature, please do the following:

- Remove all assemblies that started with ActiveReports from References.
- WebGrid.NET Enterprise 5.0 comes with newer version of ActiveReports and simpler assembly deployment for Exporting feature. Now you only need to include one assembly reference for Exporting feature.
- Add ISNet.ActiveReports.Exporting assembly from [Installation Folder]\WebGrid.NET 5.0\Bin

III. What's New in 5.0

No-codes data binding through DataSourceControl support

One of the most significant productivity gain that you can experience after migrating to WebGrid.NET Enterprise 5.0 and other 2007 product lines is the new, codeless data binding experience.

Codeless databinding means that you can configure the datasource, its connection and its related property without writing codes. In other words, you can quickly configure the WebGrid to connect to a datasource by simply using drag, drop and click.

Declaratively, you can create a WebGrid instance and bind it to a table in your Sql database in two simple statements as in the following:

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:SqlConnectionString %>"
SelectCommand="SELECT * FROM [Tasks]"></asp:SqlDataSource>

<ISWebGrid:WebGrid ID="WebGrid1" runat="server"
DataSourceID="SqlDataSource1" Height="300px"
HorizontalAlign="NotSet" Width="100%">
</ISWebGrid:WebGrid>
```

As you can see with the above statements, the WebGrid1 control is bound to a datasource control through *DataSourceID* property. The WebGrid control connects to the specified datasource control and display data from the datasource available in SqlDataSource1.

With this new databinding concept, programming databind web application is very easy, simple and straightforward. Also notice that there are no codes that required in the page's code behind. In previous version, you will need to configure InitializeDataSource event and write codes inside the event which pass the datasource to e.DataSource. That is no longer needed in WebGrid.NET Enterprise version 5.0, although the old databinding mechanism is still supported for backward compatibility.

The new datasource control binding architecture supersedes the older datasource object binding mechanism and dramatically changes the concept of how a user interface component should interact and handle datasource. To learn more about the difference and its benefits, please refer to FAQ section in the bottom of this topic.

In WebGrid.NET Enterprise 5.0, there are two major categories of datasource control support as discussed in the following sub-topics.

Basic DataSourceControl support

WebGrid.NET Enterprise 5.0 supports basic datasource control that available in ASP.NET 2.0 such as:

- AccessDataSource
- SqlDataSource
- ObjectDataSource

Three of the datasource controls above support single table view and thus can only be used as flat configuration in WebGrid control.

Advanced DataSourceControl support

In addition to basic datasource controls, WebGrid.NET Enterprise 5.0 is designed to handle more advanced datasource controls for enterprise development needs. The advanced datasource controls supported are:

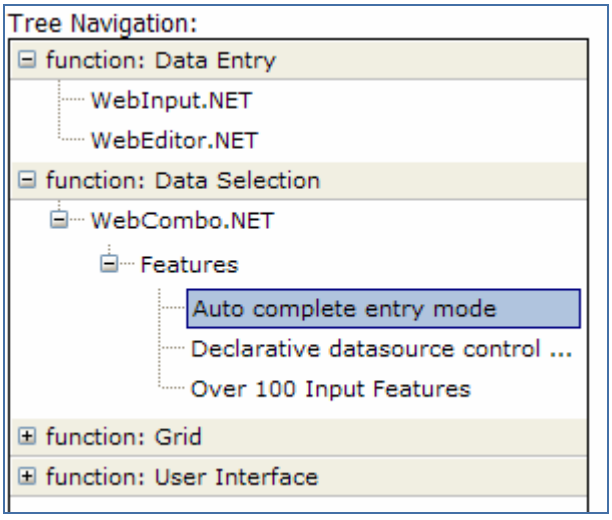
- XmlDataSource
- IntersoftDataSource

XmlDataSource control is a datasource control designed to retrieve data from xml source. WebGrid control implements a special support for XmlDataSource so that it can read the xml source defined in the XmlDataSource and display it in a tree-like layout.

Note that the XmlDataSource control support in WebGrid.NET Enterprise 5.0 is a lightweight implementation that designed to simulate a TreeView control and thus can be used only for basic display purpose. For more advanced features of TreeView control, please refer to Intersoft's WebTreeView control.

There are several limitations in WebGrid when bound to XmlDataSource control:

- Since that WebGrid utilizes its Hierarchical and ChildTable feature to simulate a tree-like display, WebGrid can effectively process structured xml only. Please avoid non-structured xml source as they will not be properly handled.
- While operating in tree-like display, most Grid-specific features will be disabled such as column sorting, filtering, load more data, paging, etc.
- Other advanced layout features are not supported and should be disabled since they are irrelevant to the TreeView function, such as columnset layout, self referencing, preview row, etc.
- The only functional feature in this mode is Grouping feature. The grouping feature enables you to group a hidden column available in the xml source. For instance, take a look at the following illustration.



To bind to an XmlDataSource control, WebGrid.NET Enterprise 5.0 comes with full design-time support that allow you to easily configure the data binding. To learn more about design-time support for XmlDataSource, please refer to Design-time support: Xml datasource control

The other advanced datasource control supported by WebGrid.NET Enterprise 5.0 is hierarchical (multi table views) datasource control that enables you to easily connect

the WebGrid control to a relational dataset available in your web application without any codes.

The hierarchical datasource control named *ISDataSource* is exclusively developed by Intersoft Solutions to support the hierarchical feature of WebGrid.NET Enterprise 5.0 while conforming the datasource control architecture offered in ASP.NET 2.0.

To learn more about ISDataSource and its advanced capabilities, please refer to ISDataSource documentation. To learn more about using ISDataSource in WebGrid, please refer to No-codes hierarchical data binding

About ISDataSource licensing in WebGrid.NET Enterprise 5.0

Intersoft DataSource Control, also known as *ISDataSource*, is sold as standalone component product that requires its own runtime license key. However, since ISDataSource plays a very important role in WebGrid's data access functionality, ISDataSource is provided at no additional cost when used together with WebGrid.NET Enterprise 5.0.

When ISDataSource detects a valid license of WebGrid.NET Enterprise 5.0 in a web application, it will be automatically functioning as licensed version. In this case, you are not required to set the runtime license key for ISDataSource in the application's web.config.

If you would like to use ISDataSource in a web application that does not contain WebGrid.NET Enterprise 5.0 (also known as standalone mode), you would need a separate license to use ISDataSource.

There are numerous standalone scenarios where ISDataSource is best used:

- Using ISDataSource for WebCombo.NET 4.0's load on demand binding.
- Using ISDataSource to replace multiple instances of *ObjectDataSource* in a web page.
- Using ISDataSource to take advantage of its advanced caching capabilities, such as file server caching and flexible caching setting.
- Using ISDataSource as the fundamental data source control for all data binding purposes. With powerful features such as automatic conflict detection, rich designers, custom object support and advanced load on demand support – it is worth to consider using ISDataSource as the main datasource control in web application development.

The benefits of the new datasource control binding support

- Improved developer's productivity via codeless, simple and hassle-free databinding.
- Cleaner separation between User Interface and Data Access/Business Layer codes. All data layer codes and logics are now handled at datasource control level instead

of component level. That means UI component such as WebGrid is now purely serving as data grid and visualization function, and no longer handling datasource.

- Better architecture. For more than decades, developer is struggling to find the best pattern in developing databound application. Datasource control binding concept is one of the most appealing and acceptable mechanism recognizing its cleaner separation between user interface and data access layer.
- Highly reusable. Datasource control offers better reusability since it has rock solid architecture. For instance, developer can write business layer codes in application level which is used throughout the application and use ObjectDataSource to connect to the business layer function.
- Extensible. Datasource control such as ObjectDataSource and ISDataSource (Intersoft's proprietary multi-views datasource control) has ability to connect to custom business layer codes. You can also connect the controls to DataSet and TableAdapter that generated using Visual Designer in Visual Studio 2005, as well as connect it to extended function that you write in an extended class through new "partial class" mechanism.

Several important notes that developer should aware with the new datasource control concept:

- *InitializeDataSource* event is no longer fired when the WebGrid is bound to *DataSourceID*. This event will still be fired normally when the *DataSourceID* property is empty (old databinding mode).
- You can't bind to datasource control and datasource object at the same time. For instance, if you set DataSourceID to SqlDataSource1 and at the same time set WebGrid1.DataSource = dsNorthWind.Customers in the code, this will throw an invalid operation exception.
- All internal operations and existing functions in WebGrid.NET Enterprise are still working properly with the new datasource control binding such as column sorting, filtering, grouping, etc. This ensures full compatibility in behaviors and functions with the previous versions.
- When the datasource control is configured for updating capabilities such as add new record, update record and delete record – WebGrid can automatically communicate to the assigned datasource control for its data manipulation process *without any codes at the Grid level*. In previous versions, developers are required to write their own codes to process the updates to physical database. For more information, refer to Walkthrough: Configuring updatable WebGrid bound to ObjectDataSource control
- Data caching. In pre-2007 platform, Intersoft's databound control such as WebCombo.NET and WebGrid.NET Enterprise employs built-in automatic data caching mechanism. In 2007 product lines, the built-in data caching is no longer functioning when the control is bound to datasource control. To configure data caching, please refer to the capability of the datasource control that you used to

bound to the control. For instance, if you bind a WebGrid control to SqlDataSource, then you can enable the caching by setting **EnableCaching** of the SqlDataSource control to True.

No-codes Hierarchical data binding

There are two types of hierarchical *select operation* supported in WebGrid.NET Enterprise 5.0 such as discussed in following sub-topics.

Load all child tables data source (default)

By default, all tables defined in the dataset that connected to the datasource control will be loaded at first page request. For instance, a dataset containing hierarchical tables such as Customers => Orders => Order Details will have all three tables loaded. This default scenario is best used to display relatively small amount of hierarchical data.

With declarative hierarchical datasource control support, displaying hierarchical data has never been easier. The following codes illustrate how the WebGrid can be declaratively connected to the datasource control which is also declaratively defined.

```
<ISWebGrid:WebGrid ID="WebGrid1" runat="server" Height="300px" UseDefaultStyle="True"
   Width="100%" DataMember="Customers" DataSourceID="ISDataSource1">
   <RootTable Caption="Customers" DataKeyField="CustomerID" DataMember="Customers">
       <ChildTables>
          <ISWebGrid:WebGridTable Caption="Orders" DataKeyField="OrderID"
          DataMember="Orders">
               <ChildTables>
                  <ISWebGrid:WebGridTable Caption="Order Details" DataMember="Order
                  Details">
                       <Columns>
                           ... columns definition
                       </Columns>
                   </ISWebGrid:WebGridTable>
               </ChildTables>
               <Columns>
                   ... columns definition
               </Columns>
           </ISWebGrid:WebGridTable>
       </ChildTables>
       <Columns>
           ...columns definitions
       </Columns>
   </RootTable>
   <LayoutSettings Hierarchical="True">
   </LayoutSettings>
</ISWebGrid:WebGrid>

<ISDataSource:ISDataSource ID="ISDataSource1" runat="server" SchemaName="dsNorthwind">
    <Tables>
           <ISDataSource:ISDataSourceTable ConflictDetection="CompareAllValues"
           SelectMethod="GetData"
           TableName="Customers" TypeName="dsNorthwindTableAdapters.CustomersTableAdapter">
           </ISDataSource:ISDataSourceTable>
           <ISDataSource:ISDataSourceTable ConflictDetection="CompareAllValues"
           SelectMethod="GetData"
           TableName="Orders" TypeName="dsNorthwindTableAdapters.OrdersTableAdapter">
           </ISDataSource:ISDataSourceTable>
           <ISDataSource:ISDataSourceTable ConflictDetection="CompareAllValues"
           SelectMethod="GetData"
```

```
            TableName="Order Details"
            TypeName="dsNorthwindTableAdapters.Order_DetailsTableAdapter">
        </ISDataSource:ISDataSourceTable>
    </Tables>
</ISDataSource:ISDataSource>
```

The ISDataSource control provides easy-to-use designer for connecting the datasource to the dataset that available in the web application. The designer has the capability to retrieve the table schemas, type name, data object methods, as well as automatic conflict detection.

Although ISDataSource control can be declaratively written in html view mode, it is best practice to configure the datasource at design time using the provided designer. The above sample has the markup generated by *ISDataSource Designer*.

To learn more about ISDataSource licensing when used together with WebGrid.NET Enterprise 5.0, see About ISDataSource licensing in WebGrid.NET Enterprise 5.0

Load child tables data source on demand

Data binding and on demand data retrieval concept has significantly changed in WebGrid.NET Enterprise 5.0 along with the new *datasource control support*.

In previous versions, WebGrid is serving as both *User Interface* and *Data Access* component. By being Data Access component, it means WebGrid also handles the datasource level functions such as initializing the datasource and performing *on demand* function on Grid level through events.

In version 5 with datasource control support, WebGrid is now serving as pure *User Interface* control. That means WebGrid no longer handle datasource level manipulation. Instead, WebGrid simply connects to a datasource control which primarily performing data retrieval and data manipulation. With this new concept, WebGrid simply displays a *valid datasource* that has been successfully retrieved by *datasource control*.

The new concept affects WebGrid's mechanism for loading child tables on demand. Several important points for loading child tables on demand in version 5.0:

- Load on demand data retrieval is no longer handled at WebGrid level.
- The load on demand data retrieval is now handled at datasource level. At the time of WebGrid.NET Enterprise 5.0 release, the only datasource control type that support this operation is *ISDataSource*.

Configuring a WebGrid to display *load on demand* child table is now easier and simpler. There is no configuration needed to set at WebGrid level.

At *ISDataSource* datasource control level, you can enable load on demand operation by simply setting *LoadOnDemand* property to True. Next, you provide the method that retrieve the child table data and assign it to the *ISDataSource* control.

```
<ISDataSource:ISDataSource ID="ISDataSource1" runat="server"
    LoadOnDemand="True" SchemaName="dsNorthwind">
...
</ISDataSource:ISDataSource>
```

There are two ways to provide the method for retrieving child table data:

- Write codes for the method in *App_Code*.
  The following codes illustrate the technique to load *Orders* table when a *Customer* record is drilled down.

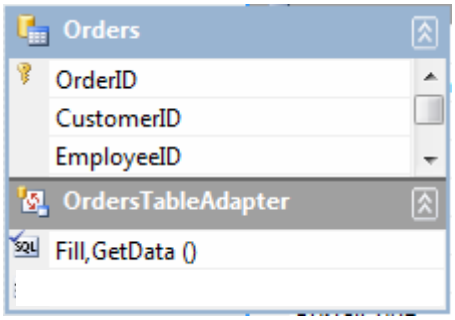
```
public DataTable GetData(string customerID)
{
      dsNorthwind.OrdersDataTable dt = new
      dsNorthwind.OrdersDataTable();

      OleDbDataAdapter adapter = new OleDbDataAdapter();

      adapter.SelectCommand = new
      OleDbCommand("SELECT * FROM Orders WHERE CustomerID =
            '" + customerID + "'", this.Connection);
            adapter.Fill(dt);

      return dt;
}
```

  Using this technique, you can assign the ISDataSourceTable's *SelectMethod* to *GetData*.

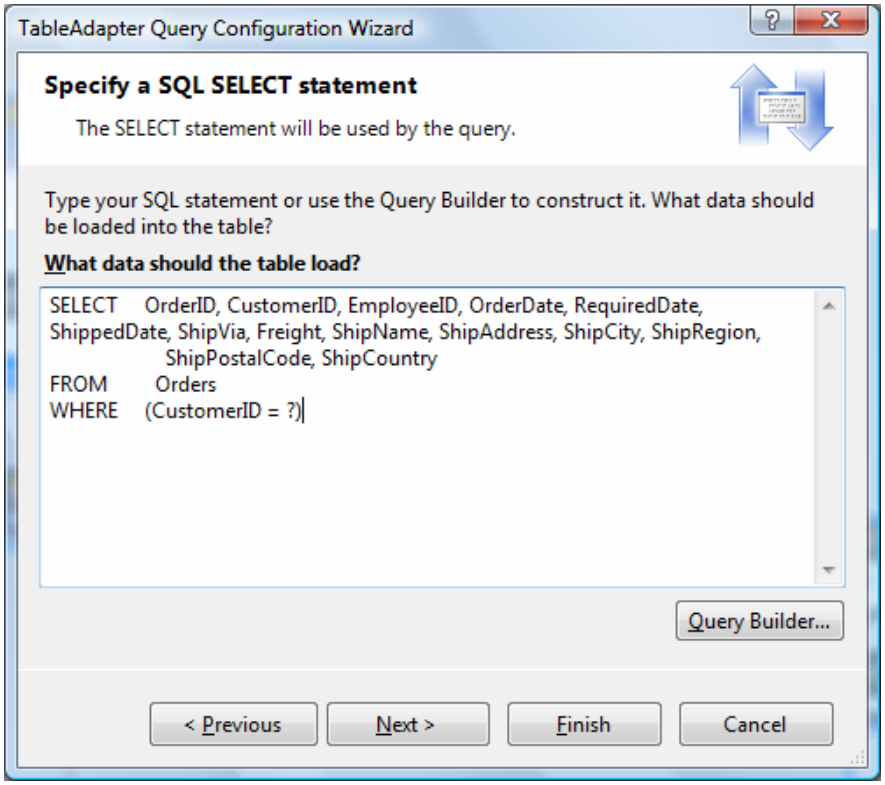
- Create the method in Visual Studio 2005's *DataSet Designer*.
  Another way to create the data method is through *DataSet Designer* in Visual Studio 2005. The *DataSet Designer* enables you to create the method using a visual manner with less coding needed.

  See following steps for more details:

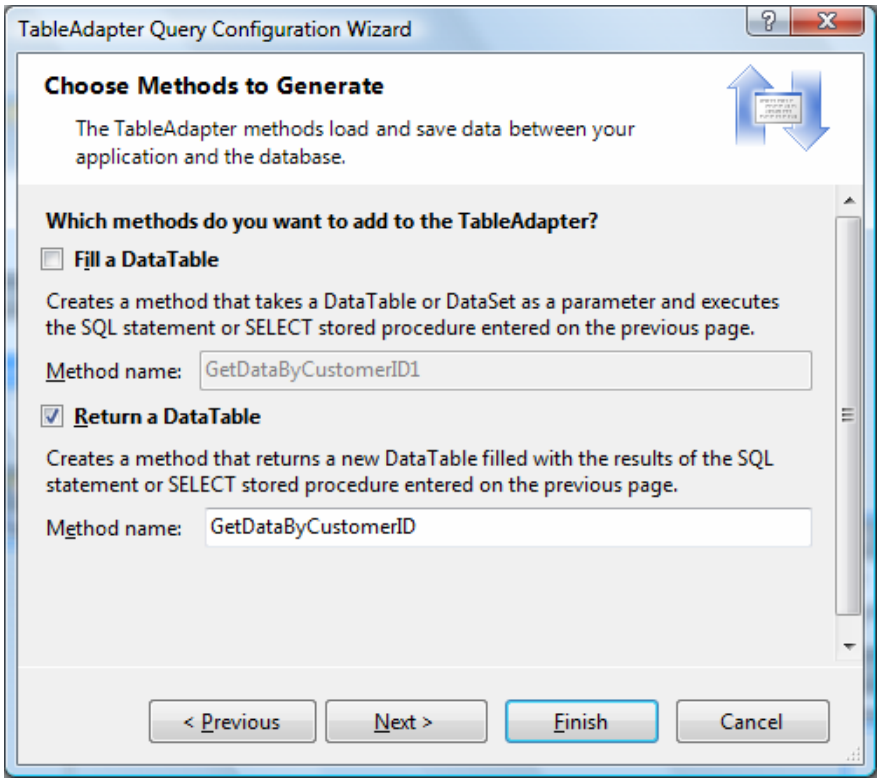
  - Firstly, right click on the *TableAdapter* area and click "Add Query".



  - Next, you can write the SQL statement or use Query Builder to select the data to be used by the *datasource control*.



- Next, name the method as "*GetDataByCustomerID*" in the *Returns a DataTable* option. Uncheck the *Fill a DataTable* option. Notice that the *CustomerID* parameter will be added by the designer automatically.



After the method is successfully created, you can simply assign it to be consumed by *ISDataSource* by setting the *SelectChildRowMethod* of *Orders* DataSourceTable to *GetDataByCustomerID*. Note that in this technique, the *SelectMethod* is still set to *GetData*. The *SelectChildRowMethod* is provided for the convenience in setting the child row data method without has to change the main select method.

Once the ISDataSource control has been successfully configured, WebGrid will be able to automatically perform load on demand data retrieval without additional settings or codes at WebGrid level.

No-codes WebValueList data binding

With declarative datasource control support in version 5.0, WebGrid makes it very easy for developers to bind the control to a datasource. In addition to binding the Grid to datasource control for its main data, WebGrid also enables you to bind a column's value list to datasource control. This feature completely removes the needs to write codes in the Grid control and allow you to declaratively specify the datasource control and datamember for the value list.

As an example, the value list for a *WebGridColumn* can now be specified declaratively such as:

<ValueList DataMember="Products" DataTextField="ProductName" DataValueField="ProductId" />

In previous versions, the only way to setup *WebValueList* is through code behind. You will need to write several line of codes to populate a *WebValueList*. The declarative WebValueList feature in version 5.0 significantly improves developer's productivity by eliminating codes.

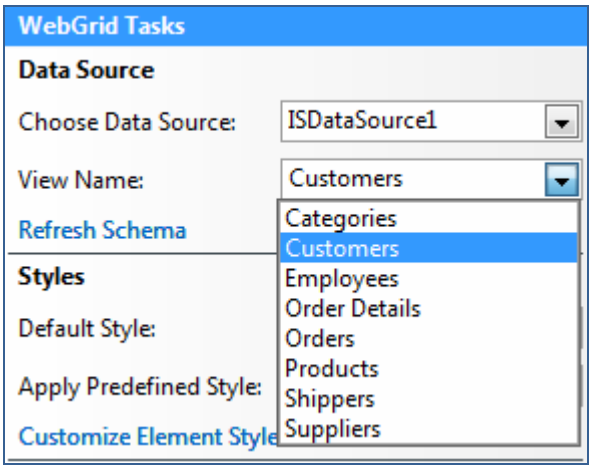
When the WebGrid is bound to ISDataSource control, you can bind the *WebValueList* to an existing *ISDataSourceTable* defined in the ISDataSource control. In this case, you don't need to specify the *DataSourceControlID* of the *WebValueList*, such as shown in the above markup sample.

Design-time support for DataSourceControl

WebGrid.NET Enterprise 5.0 boost developer's productivity by introducing various ways to connect to a *datasource control* in design-time through rich designers and wizards.

SmartTag Action Panel

The easiest way to start with data binding is through *SmartTag Action Panel*. By default, when you drop a new instance of WebGrid to designer, the SmartTag will appear automatically as shown in the following image.

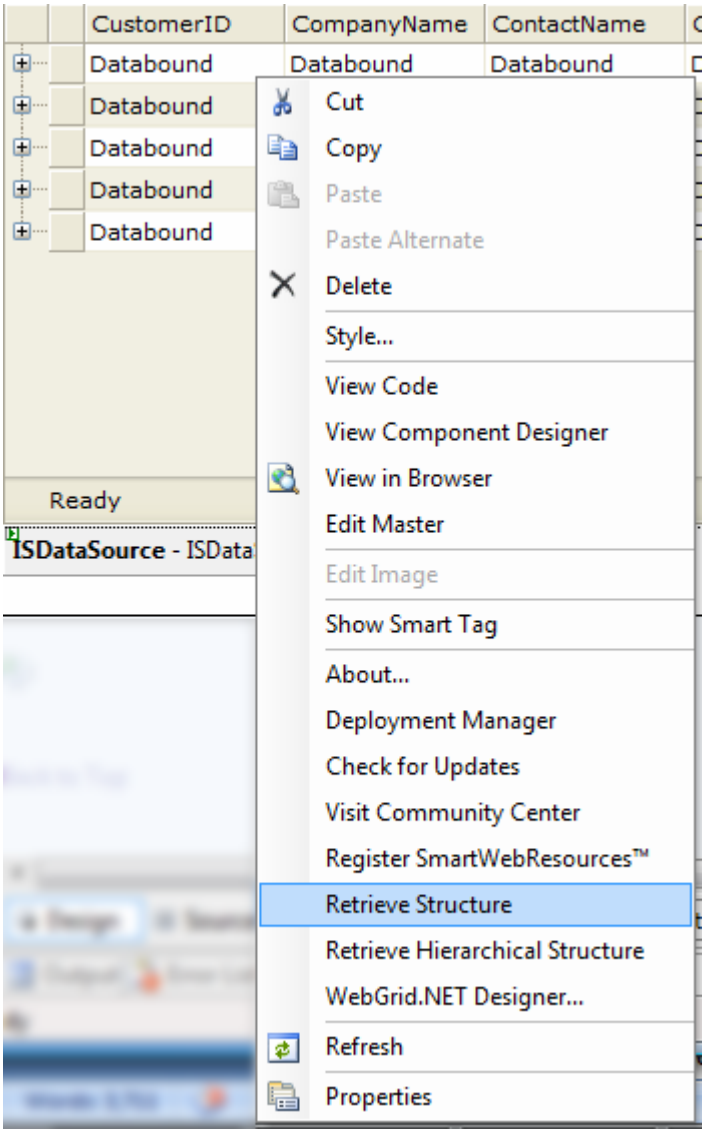


The Choose Data Source dropdown enables you to quickly connect to a datasource control available in the web page. The dropdown list will list all valid datasource controls. You can also click on *<New data source…>* item to create a new datasource control from scratch. This feature is reflected directly to *DataSourceControlID* property of WebGrid.

When connected to a *datasource control* that support multiple views such as in *ISDataSource* control, the *View Names* dropdown will list available views defined in the specified datasource control such as shown in above image. This feature is reflected directly to *DataMember* property of WebGrid.

Retrieve Structure and Hierarchical Structure Context Command

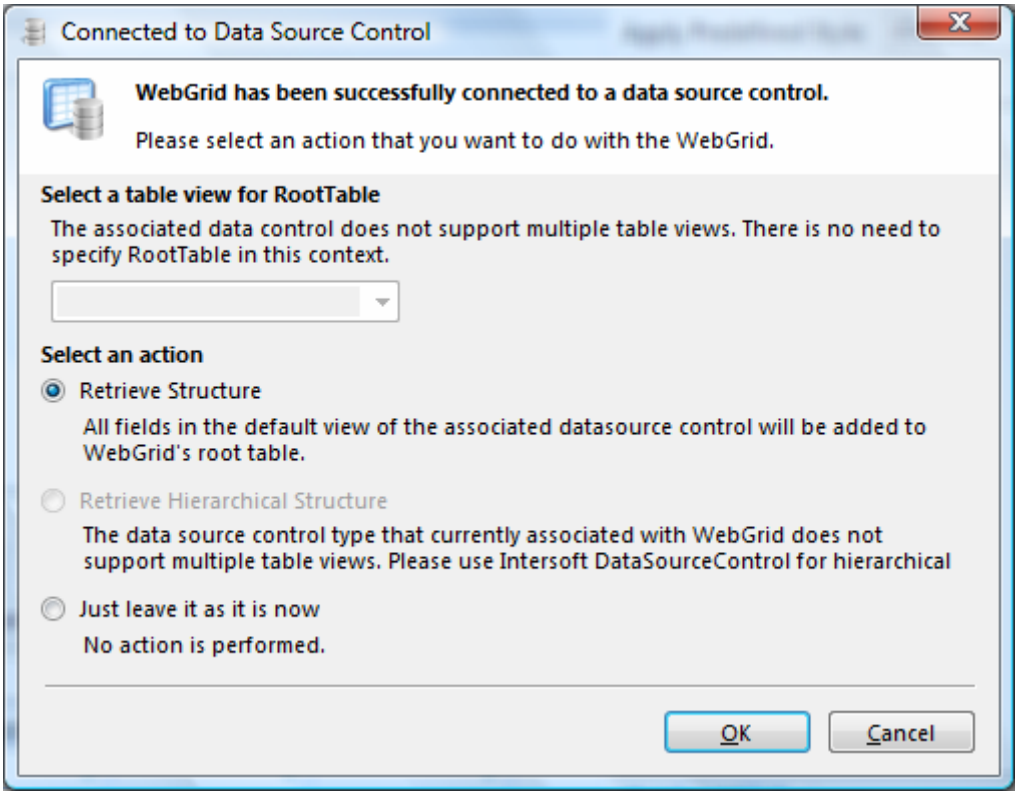
When you have modification to datasource schema which already assigned to WebGrid, you don't need to change the *DataSourceControlID* or *DataMember* property. Instead, you can simply re-retrieve the schema (structure) by using the commands in control's context menu. Refer to the following image for details.



Note that if the existing WebGrid already has structures defined such as Columns, GroupColumn, SortColumns etc – the *Retrieve Structure* and *Retrieve Hierarchical Structure* will override existing structure with new schema defined in the datasource control.

Flat table datasource control

When the WebGrid is connected to flat datasource control, such as *AccessDataSource* or *SqlDataSource* control, the *WebGrid Datasource Connect* wizard dialog box will appear, such as illustrated in the following image.



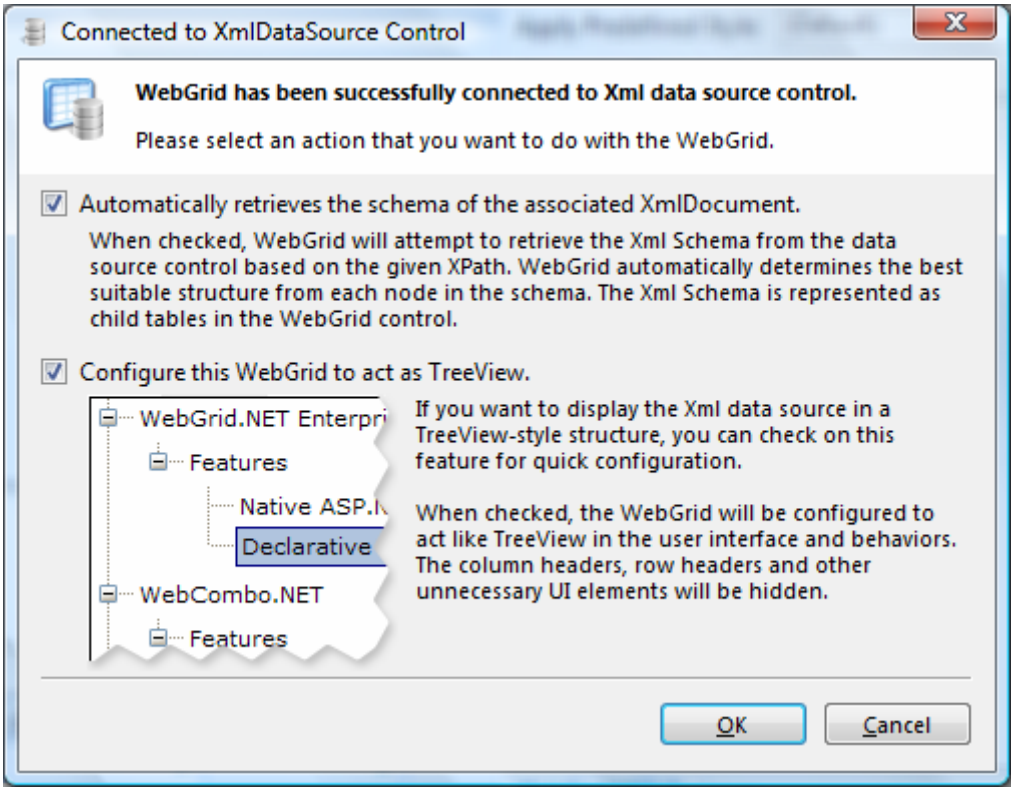
The datasource connection wizard integrates deeply with Visual Studio 2005® to provide rich design-time experience, such as automatic datasource detection when the *DataSourceID* property has changed. The wizard also has ability to retrieve the datasource schema available in the datasource control.

As seen in above image, the *view names* dropdown list and *retrieve hierarchical structure* option are automatically disabled. In flat-view datasource control, the datamember does not need to be set as the datasource control should contain only 1 table view. The flat table view is also called as *DefaultView* data member.

Xml datasource control

WebGrid.NET Enterprise 5.0 exclusively supports XmlDataSource control and display the Xml data as hierarchical Grid to simulate TreeView. For more information about XmlDataSource control support in WebGrid.NET Enterprise 5.0, see Advanced DataSourceControl support.

When connected to an *XmlDataSource* control in design-time through either SmartTag Action Panel or Property Window, the *XmlDataSource Connect* wizard will be displayed as shown in following image.



The XmlDataSource Connect wizard has two primary features:

- Auto retrieve schema.
  WebGrid simulates the TreeView display by using its Hierarchical and ChildTables feature. This option allows WebGrid to inspect the XmlDataSource schema, read and translate it to nested ChildTables in WebGrid control.

  It is recommended to check this option as default.

- Auto configure as TreeView.
  WebGrid provides many settings and flexibility to customize appearance. With this option checked, WebGrid automatically configures its appearance so it looks like a TreeView control.

Several notes related to XmlDataSource control support:

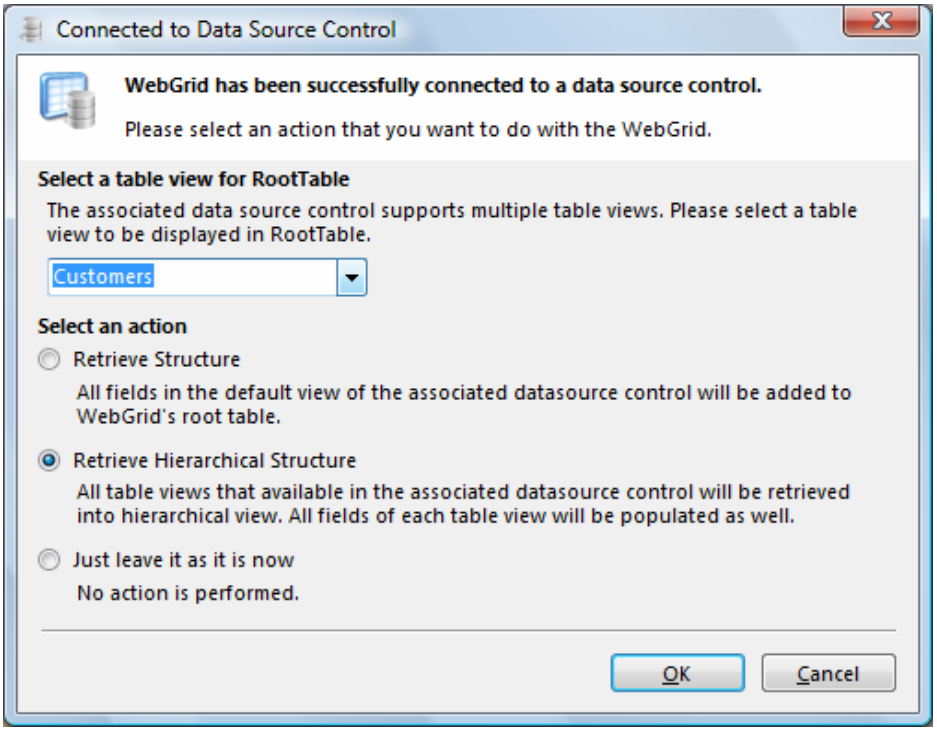
- The TreeView support in WebGrid.NET Enterprise 5.0 is provided as basic complimentary feature to datasource control only. It is not intended to be as used as TreeView's replacement. For more advanced TreeView-specific functions, please use *Intersoft's WebTreeView for ASP.NET 2.0* instead.
- WebGrid simulates the TreeView support by loading all Xml Data at first load. In this case, all Tree nodes are rendered at once in first request and no load-on-demand is necessary. Simply put, the WebGrid is working as Unbound mode.

- With its Unbound mode behavior limitation, data-related features – such as Sorting, Filtering, Paging, Editing and others – are not supported and should be disabled. However, Grouping is exclusively supported, and can be enabled.
- WebGrid is treating each XmlAttribute of a Node as Column. The auto-generated schema always assume first attribute to be used as display text in the Tree. To change the column to be used as display text, simply move the designated column to first column and set its Visible property to True. Other columns Visible should be set to False. You may also want to set the first column's width to 100% to properly simulate TreeView display.

Hierarchical datasource control

Similar to flat-view datasource control and XmlDataSource control, WebGrid automatically displays *DataSource Connect* wizard when a valid multi-view (hierarchical) datasource control is assigned to WebGrid.

The following image shows the wizard for Hierarchical datasource control connection.



When connected to a valid hierarchical datasource control, WebGrid allows you to perform the following actions in designer:

- Select a view available in the *View Names* dropdown as WebGrid's RootTable. The selection of the view in RootTable determines and affects the hierarchical child tables that will be automatically retrieved.
- Retrieve hierarchical structure. This option is enabled when WebGrid detects a valid hierarchical datasource control being assigned. When selected, WebGrid

automatically retrieves the child tables based on the *relationship definition* that specified in the associated dataset. The child table retrieval starts from the table view that assigned as RootTable (see option 1 above).

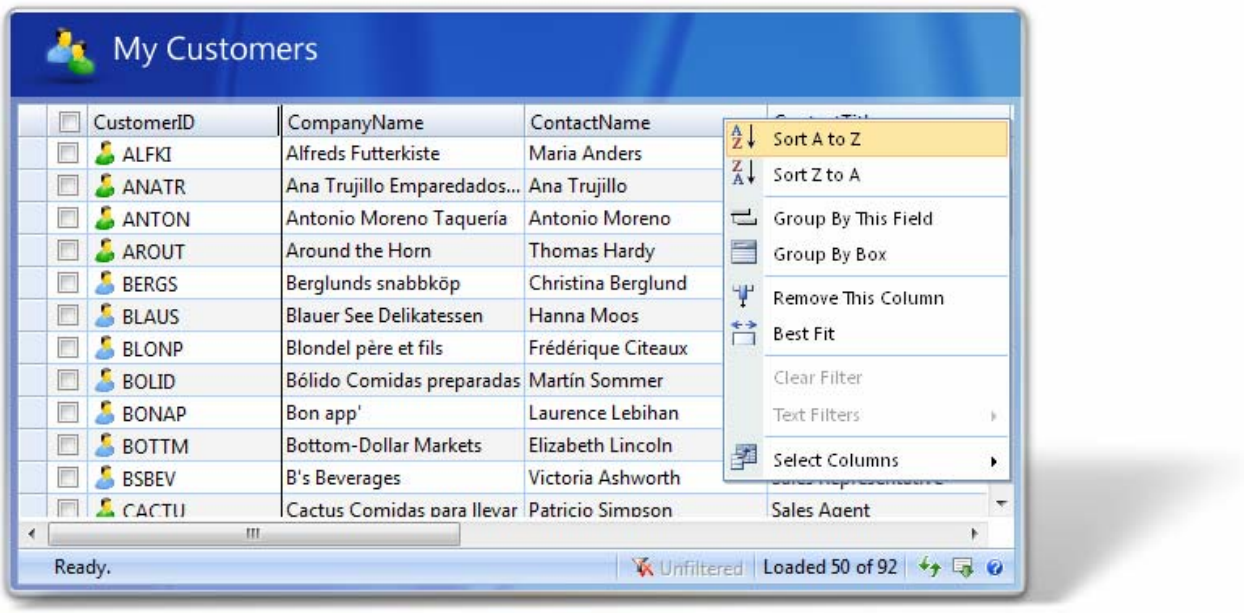
In order for Retrieve Hierarchical Structure to work successfully, ensure you have defined the table's relationship properly using *DataSet Designer*.

New User Interface, New Styles

WebGrid.NET Enterprise 5.0 is the most significant release since its initial debut. Version 5.0 includes many enhancements to Grid-specific functions, data binding and new features in *User Interface*.

WebGrid.NET Enterprise 5.0 comes with new user interface such as new icon-sets, new context menu engine and styles, as well as new "Elegant" style. These new features help end users to interact with information better with a Grid which is not only powerful in functions, but appealing to eyes and experience as well.

The following screenshot captures the "*First Experience*" sample of WebGrid.NET Enterprise 5.0. The new visual appearance and interface redefines user experience while interacting with the control.

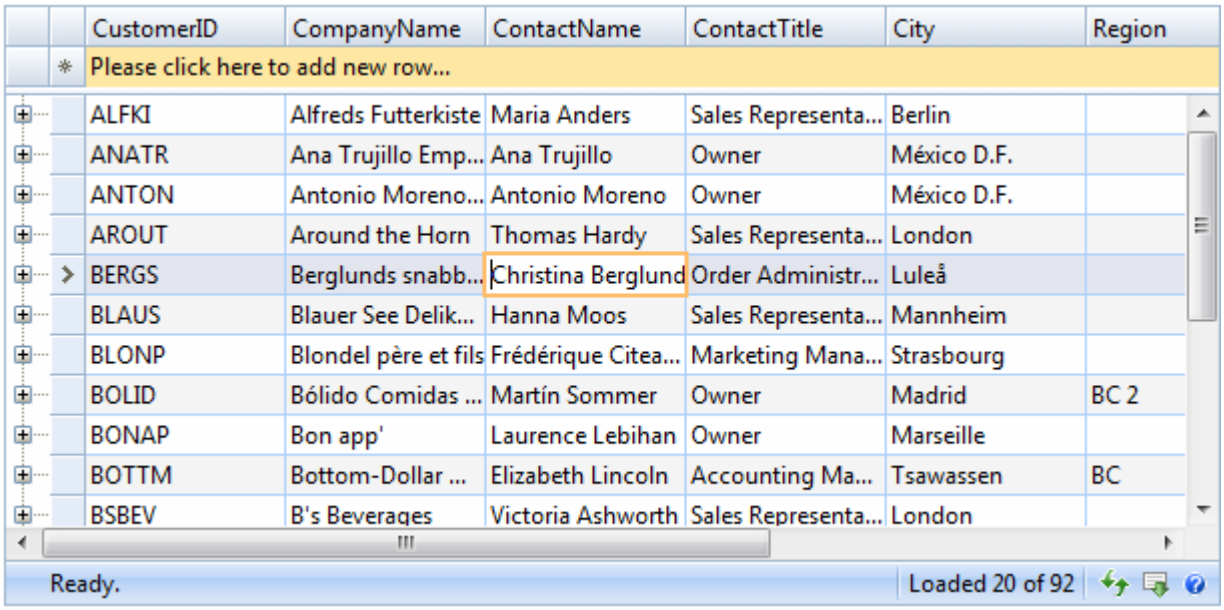


Some of the new features related to user interface are:

- Status bar group. The status text is now easier to be read with clear separation between each different function group.
- New theme and improved styles with smooth blue "Glass" appearance in *Elegant* style.

- New set of icons, such as new status bar icons and better loading indicator. All context menu items are also having new icons as well.
- User-friendly command messages based on data type. For instance, sorting text is now customized according to the data type. These new textual elements are included as localizable text (controlled by Localization feature).
- New *RowHighlightType* for more convenient editing experience.
- New high-performance context menu interface with smooth fading animation and shadow effect.
- Improved User Interface for Error Dialog Box.
- Visual indicator for action status and editing information in Status Bar.

The following screenshot demonstrates the new "Elegant" style while operating in editing mode. Notice that cell border of selected row remains visible (new *RowHighlightType*) and larger edit cell for more convenience editing experience.



Column Freezing

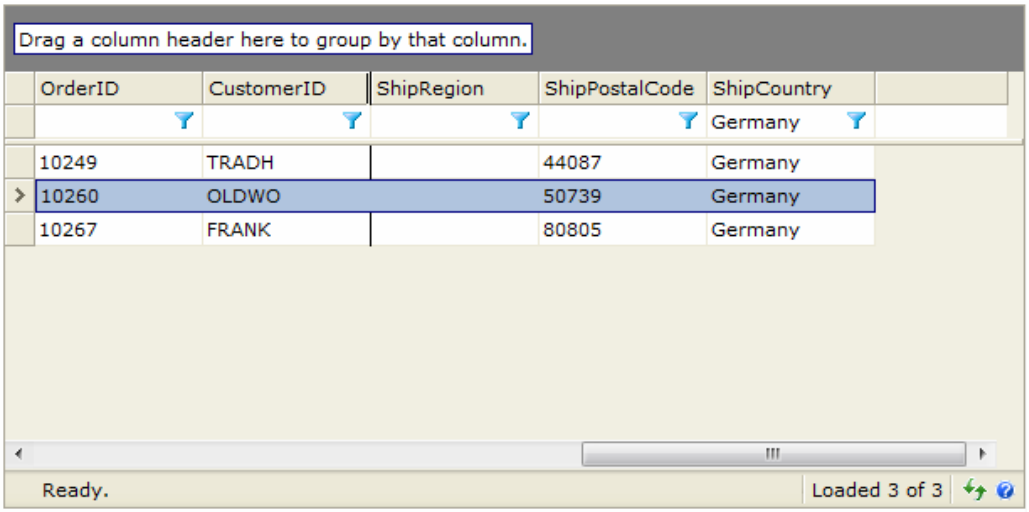
WebGrid.NET Enterprise® 5.0 is the industry's first WebGrid that successfully implementing Microsoft Excel® style *column freezing* feature. In this new version, WebGrid introduces *LiveFreeze™* technology which enables users to *"lively"* perform column freezing and unfreezing in realtime without the need of postback or callback.

WebGrid's unique *LiveFreeze™* architecture enables frozen columns to stay visible eventhough the horizontal scrollbar of the WebGrid is scrolled. Some of great results from *LiveFreeze™* implementation are:

- High performance scrolling. The Grid is capable to handle fast scrolling by either dragging the scrollbar's thumb or click on the left and right arrow.

- Excel® scrolling behavior. When scrolled, hidden columns must be entirely hidden (instead of partially).
- Splitter indicator at column header and cell to let users easily determine the current state of frozen columns.
- Works with all existing WebGrid's functions and FlyPostBack (AJAX) actions such as Grouping, Sorting and Filtering. Try to perform sorting or grouping, notice that scroll position will stay and hidden columns will remain hidden.
- "Lively" perform freezing and unfreezing in realtime. No AJAX callback or postback is required.
- Works perfectly with keyboard navigation in cell select mode as well as in editing environment. The hidden column will be automatically visible and focused during editing, the scrollbar will also automatically reflect the current position of the edit cell.
- Column moving and resizing is supported for unfrozen columns. When moved or resized, scrollbar and current view will be automatically adjusted.

The following image shows a simple WebGrid with *Column Freezing* feature enabled. The *OrderID* and *CustomerID* column remains visible, while the horizontal scrollbar is set to the right most position.



Main Concept

LiveFreeze™ enables you to configure initial column freezing setting from server side property. There are two main properties that need to be set to activate this feature: *AllowColumnFreezing* property in *LayoutSettings* and *ActiveFrozenColumns* property in *FreezePaneSettings*.

The *ActiveFrozenColumns* property defines how many columns are frozen in initial page load. The default value is 0, which indicates there are no frozen columns that need to be configured. The value of *ActiveFrozenColumns* property determines the number of visible columns to be frozen in respective columns order. For instance, if you have a WebGrid with columns "FirstName, LastName, ContactTitle" and LastName's *Visible* is

False and *ActiveFrozenColumns* set to *2*, then the active frozen columns would be "FirstName" and "ContactTitle".

It is best practice to keep the *ActiveFrozenColumns* property to a value where the column freezing feature may deliver best results in user experience. For instance, if you set *ActiveFrozenColumns* to a value exceeding half of the columns length, then the column freezing display might not be useful.

Most settings related to column freezing are represented in *FreezePaneSettings* object, which is located inside *LayoutSettings*. You can also configure the *MaxFrozenColumns*, the splitter's line color and style and much more.

The above image shows a WebGrid with *ActiveFrozenColumns* property set to *2*.

Runtime freezing/unfreezing

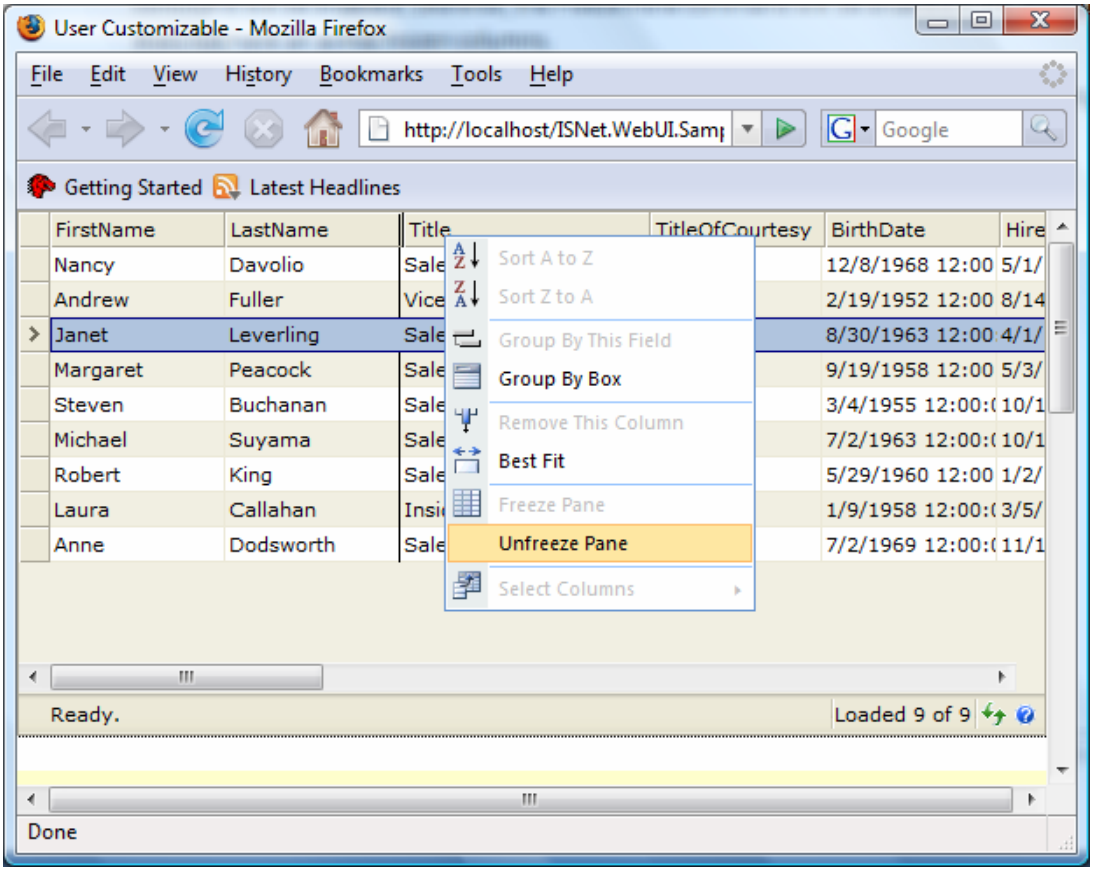
WebGrid's LiveFreeze™ feature allows users to perform freezing/unfreezing at runtime through column context menu. When the WebGrid detects an active frozen settings, the *Unfreeze Pane* command will be enabled. Likewise, the *Freeze Pane* command will be enabled if the WebGrid does not have an active frozen columns.

The runtime freezing feature can be easily enabled by setting *ShowInContextMenu* property in *FreezePaneSettings* object to True.

For example:

```
<LayoutSettings AllowColumnFreezing="Yes"
        AllowColumnMove="Yes">
      <FreezePaneSettings ActiveFrozenColumns="2"
       MaxFrozenColumns="3" ShowInContextMenu="True" />
</LayoutSettings>
```

The following image shows the result of the above sample in Mozilla Firefox 2.0 browser:



Absolute Scrolling

To better simulate Excel® scrolling behavior while freezing is active, WebGrid.NET Enterprise® 5.0 introduces Absolute Scrolling™ in conjunction with the LiveFreeze™ implementation.

With *Absolute Scrolling* enabled, WebGrid is able to automatically adjust the viewport to the next available column when user pressed on the left/right arrow of horizontal scrollbar. This behavior allows users to conveniently scroll through all the columns with less clicks.

This feature is controlled through *AbsoluteScrolling* property in *FreezePaneSettings* object. To learn more, see How-To: Enable absolute scrolling

LiveFreeze™ behaviors and other notes

The following is a list of helpful notes related to LiveFreeze™ behaviors when used with other WebGrid's features:

- Frozen columns are designed for flat-view and can be applied to RootTable only. Child tables are not applicable.
- Frozen columns are not moveable. This behavior is set automatically.
- Frozen columns will not appear in *Select Columns* automatically.

- Frozen columns can't be grouped, unless *HideColumnsWhenGrouped* property is set to *False*.
- Frozen columns visibility are locked and can't be set to hidden. If you would like to enable users to remove the column, unfreeze the column first to remove it from the Grid.
- Frozen columns can't work together with *ColumnSet* layout, since *ColumnSet* layout does not have suitable display and purpose to be applicable for Column Freezing feature.
- Column Freezing feature should not be used with *AutoFitColumns* feature due to purpose's conflict. AutoFitColumns feature purpose is to prevent horizontal scrolling by auto-fitting all columns, while Column Freezing feature requires horizontal scrolling.
- Column Freezing feature is designed for flat-view Grid. However, you can still enable *Hierarchical* Grid to be used together with this feature.
- When *Hierarchical* is enabled along with Column Freezing feature, horizontal scrolling affects to RootTable view only. The child table will not be scrolled.

  Design Considerations: Enable *AutoFitColumns* at child table level so that the columns fit into the main Grid's viewport. Arrange the child table layout so they are always inside the Grid's boundary to keep all columns visible.
- For best performance, limit the *RootTable* rows to a reasonable maximum number of rows. Enable *VirtualLoad* paging if needed.
- If you are displaying large data (eg, more than 1000 rows) in the WebGrid, it's recommended to set *ShowSplitterLine* to *false* for best performance.
- For best usability, set the *MaxFrozenColumns* to appropriate value where the frozen columns can be keep visible even in minimum window size.
- When the frozen pane's width exceeds the visible client area of the WebGrid, the horizontal scrolling will automatically fallback to *default scrolling behavior*. This means the frozen pane will no longer in effect until the frozen pane's boundary is visible back in the client's viewport. This automatic behavior ensures WebGrid to behave and display properly as users working with information.
- When column freezing is enabled, the horizontal scrollbar will always appear regardless of the WebGrid's width. This is designed for consistent user interface.
- When column freezing is enabled, you can still access the client side Columns object model as usual since LiveFreeze™ used a new developed algorithm that does not require changes in core DOM structure.
- Hidden columns that resulted from the scrolling process will be automatically persisted in FlyPostBack™ (AJAX) actions, except *Refresh* and *RefreshAll* action. The hidden columns will not be persisted in full postback event.
- Column freezing feature can be used together with the following features:
  - Column resize and moving (for non-frozen columns)
  - Grouping

- Filtering
- Sorting
- GroupTotal
- AllowAdd
- AllowEditing
- Keyboard Navigation
- VirtualLoad™ Paging
- CellSelect mode
- EditOnClick mode
- PreviewRow

o Extensively supported in Mozilla browsers. Optimized for latest browsers such as Internet Explorer 7.0+ and Mozilla FireFox 2.0+

Cleaner and reduced page output through built-in Default Style

New in version 5.0 is the default style concept to deliver optimal performance right out-of-the-box.

The default style feature offers the following benefits:

- Clean control definition during development. This significantly improves loading time in the Visual Studio 2005 designer because unnecessary style objects do not need to be created. For instance, WebGrid.NET Enterprise version 5.0 only requires the following simple tag.

```
<ISWebGrid:WebGrid ID="WebGrid1" runat="server"
UseDefaultStyle="true">
</ISWeGrid:WebGrid>
```

  In previous version, WebGrid.NET Enterprise automatically generates a lot of codes containing the styles definition in order for the WebGrid control to display properly.

- Best performance in server throughput. If your end users are satisfied with the default style of WebGrid, we highly recommend you to use the default style whenever possible. By enabling default style, the server side processing is significantly reduced because there is no need for serializing styles, validating and parsing the styles and finally generate the styles into css-based classes in the page output.

- Reduced page output size up to 70 percent. Based on a research, more than 60 percent of web developers do not modify the styles of Intersoft's controls in their web application.

  In previous version of WebGrid, all styles are automatically generated to the page output, regardless whether the style is the one that come with the control or custom style that applied by the developer. So let's say that you have 10 instances of WebGrid that use default style, the control will generate each style of each instance to the page output. This is normal behavior in previous version since there is no mechanism that tell the control whether the defined style is a default style or custom style.

In version 5.0 with the Default Style feature, WebGrid control no longer produces page-level styles for instances that enable Default Style. This mechanism enables page output reduction up to 70 percent, since styles are no longer produced regardless of the number of WebGrid instances in a page.

The following illustration compares the page output between WebGrid.NET Enterprise 5.0 and its predecessors.

WebGrid.NET 1.x – 4.x	WebGrid.NET Enterprise® 5.0
<style> . WebGrid1-Row { … } . WebGrid1-Header { … } . WebGrid1-Frame { … } … . WebGrid1-Row { … } . WebGrid1-Header { … } . WebGrid1-Frame { … } </style> Total bytes occupied for style per control: ± 20 KB	<style> /* empty */ </style> <link rel="stylesheet" href="/WebApp/WebResource.axd?d=x &t=632936740395297500" type="text/css" /> Total bytes occupied for style : 0 KB External style sheet size for default style: 2 KB (reusable and cacheable for other pages)

Several important things to note around Default Style feature:

- When Default Style is enabled, WebCombo will include a link to external stylesheet that contains reusable style definition used by the Default Style. This external resource is fetched by using standard WebResource feature, and is automatically handled by WebCombo. There is no efforts required by developers.
- When Default Style is enabled, all styles definition in the control instance should be removed. You can't have styles definition and using default style at the same time.
- Default Style feature is enabled by default for new instance of WebCombo.NET 4.0. Exisiting instances of WebCombo (i.e., after migrated from previous version) will continue to use their existing styles definition.

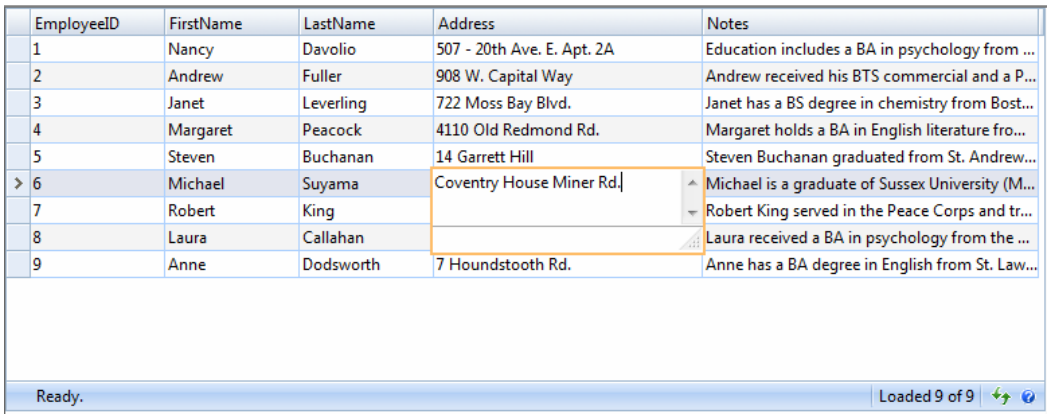
Default Style Mode

WebGrid.NET Enterprise 5.0 comes with two default styles as discussed in the following:

- Standard Style
  This is the classic WebGrid style which has been improved to use more neutral colors. By default, Standard Style is used when the DefaultStyle feature is enabled.
- Elegant Style

This is a new style introduced in WebGrid.NET Enterprise 5.0. Included as *DefaultStyle*, you can easily change the WebGrid's appearance to use this new style by simply setting the *DefaultStyleMode* to *ElegantStyle*.

The *ElegantStyle* mode sports modern visual such as glass backgrounds combined with soft blue backgrounds. The new *ElegantStyle* is designed to deliver visually compelling user interface and better user experience. See the following image for more details.



EmployeeID	FirstName	LastName	Address	Notes
1	Nancy	Davolio	507 - 20th Ave. E. Apt. 2A	Education includes a BA in psychology from ...
2	Andrew	Fuller	908 W. Capital Way	Andrew received his BTS commercial and a P...
3	Janet	Leverling	722 Moss Bay Blvd.	Janet has a BS degree in chemistry from Bost...
4	Margaret	Peacock	4110 Old Redmond Rd.	Margaret holds a BA in English literature fro...
5	Steven	Buchanan	14 Garrett Hill	Steven Buchanan graduated from St. Andrew...
6	Michael	Suyama	Coventry House Miner Rd.	Michael is a graduate of Sussex University (M...
7	Robert	King		Robert King served in the Peace Corps and tr...
8	Laura	Callahan		Laura received a BA in psychology from the ...
9	Anne	Dodsworth	7 Houndstooth Rd.	Anne has a BA degree in English from St. Law...

Ready. Loaded 9 of 9

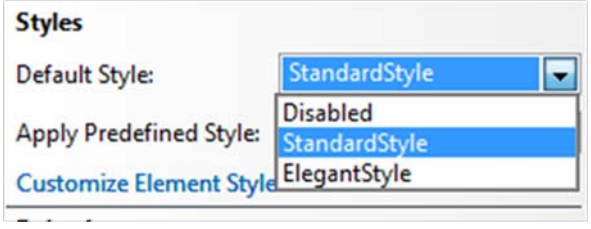
FAQ: What does Default Style look like?

Please refer to New User Interface topic to see the look and feel of the Default Style.

How-to: Enable Default Style for existing instance of WebGrid

You can enable default style in one of the three way described in the following:

1) SmartTag Designer.
   The easiest way to enable Default Style for an existing instance of WebGrid is through SmartTag Designer. SmartTag Designer is a new feature in WebGrid.NET version 5.0.
   To enable default style from SmartTag Designer:
   - Select an instance of WebGrid.
   - If the SmartTag Designer Panel does not appear, please click the ▶ symbol that appear to the right top relatively to the WebGrid.
   - With the SmartTag Designer Panel displayed, click on the "Default Style" dropdown list which is under Styles category.



Using SmartTag Designer is the recommended way to enable Default Style feature. The Default Style selection will automatically clean up all style tags in the control definition.

2) Property Set.
You can also manually enable the Default Style feature by setting the *UseDefaultStyle* property to True in the Visual Studio's property window.

However note that this manual approach does not automatically clean up existing style definition.

Hassle-free deployment through SmartWebResources™ technology

Overview

Oftentimes, developers found it too complex in order to deploy their web application that used third party components. There are too much things that need to be recalled before a web application can be successfully deployed. For an instance, the javascript files, images and other runtime client files need to be copied in order for the component to run properly.

One of the goal in Intersoft's 2007 product platform is to simplify the deployment process required in Intersoft's controls. Our approach is by developing a new infrastructure that handles the storage and retrieval system of these client resources so that physical files can be totally eliminated. This new infrastructure is called SmartWebResources™.

WebGrid.NET Enterprise version 5.0 fully supports SmartWebResources infrastructure. This enables WebGrid control to function perfectly right out-of-the-box after the simple XCOPY deployment.

Both developers and site administrators can now deploy Intersoft's components by simply copying the assemblies in the Bin folder of the project – and totally eliminates the need to copy javascript files, component-specific images and runtime files.

As best practice, it is highly recommended to enable SmartWebResources feature for all Intersoft's components especially WebGrid. Furthermore, WebGrid.NET Enterprise 5.0 exclusively requires *WebDesktop's SmartWebResources Assembly* in order to use some advanced *User Interface* features in WebGrid such as the new ContextMenu engine and runtime engines for dialog boxes and windowing system.

To learn how SmartWebResources works, please read How does SmartWebResources™ work

To learn more about the benefits introduced by SmartWebResources, read Features and benefits introduced by SmartWebResources™

About Localization Files

WebGrid.NET Enterprise includes numerous language files in Xml Format for localization purpose. In previous version, the language files must be located inside IIS directory to be accessible from client.

In version 5.0, language files are included in *WebGrid SmartWebResources Assembly*. This means the physical language files in IIS directory are no longer required.

If the language file is set to be retrieved from *SmartWebResources*, you can't modify or customize the texts (strings) as the files are now embedded inside assembly. If you want to use modified version of language file, then you can switch the language retrieval back to classic physical files retrieval mode. For more information, see How-to: Disable SmartWebResources for Localization's language files

FAQ: Does SmartWebResources feature automatically enabled in WebGrid.NET Enterprise 5.0?

By default, Intersoft 2007 product platform have the *SmartWebResources* set to *Automatic* mode. This automatic mode ensures compatibility with existing web application when developers migrated to 2007 platform, while at the same time attempting to utilize SmartWebResources in new web application whenever possible.

When you created a new web application and started to use WebGrid.NET version 5.0, the SmartWebResources is not configured initially. With the automatic mode as default behavior, WebCombo.NET will use physical resources since the SmartWebResources is not configured initially.

FAQ: Briefly, what are the files included in WebGrid SmartWebResources Assembly?

The single assembly of WebGrid's SmartWebResources contains all required scripts and images that used by the control at runtime. It includes:

- Core Scripts, such as WebGrid_Core.js, WebGrid_UI.js etc.
- Custom Editor Scripts, such as the scripts for Slider, Rich Html Editor etc.
- Core Images Set, such as tree lines, background images, command icons etc.
- Theme Images Set, such as the Outlook-style images used in *Outlook Predefined Style*.
- Language Files, such as the Xml Data for English language. There are about 15 languages included in this resource assembly.
- Stylesheets and other document files that will be automatically loaded when needed by the module in client side.

FAQ: I got a warning message "ContextMenu is disabled due to missing resources" after page load in the browser. What should I do to resolve this problem?

If you get this warning prompt, that means the WebGrid is unable to find the *WebDesktop SmartWebResources Assembly* in runtime. Ensure you have properly registered the SmartWebResources.

FAQ: I am aware that WebGrid.NET Enterprise 5.0 requiring WebDesktop SmartWebResources as part of its User Interface engine. Do I need to purchase a separate WebDesktop's license in order to use its SmartWebResources?

No, you do not need to purchase a separate license for WebDesktop to use its *SmartWebResources*. However, you may use and redistribute *WebDesktop SmartWebResources Assembly* only when you use it together with WebGrid.NET Enterprise 5.0.

Note that starting from 2007 product platform, all products are installed through a single integrated installer. This enables deeper integration between Intersoft's components in runtime without affecting licensing. For more information about new installation and licensing mechanism in 2007 product platform, refer to WebUI.NET Framework 2007 Documentation.

How-to: Configure SmartWebResources in a new web application

As the best practice in using Intersoft 2007's product lines, you are recommended to perform the following steps after you have created a new web application or migrated an existing web application.

1) Right click on one of the Intersoft's control instance. For instance, WebGrid1.
2) Click Register SmartWebResources™ command in the context menu.

3) In short time after you clicked the command, you should receive a message prompt similar to the following:
"SmartWebResources has been successfully configured for this web application."

If you are receiving an error message, it is likely that you have not installed the product properly. Please note that you can't enable this feature in older products (non 2007-platform products).

To learn more about SmartWebResources, please read Introducing SmartWebResources™

How-to: Disable SmartWebResources for Localization's language files

You can specifically disable the *SmartWebResources* for language files only without affecting *SmartWebResources* for scripts and other client resources.

To disable the SmartWebResources for language files, set *UseWebResources* of *TextSettings* object to *false*. When disabled, the language file will be retrieved using the classic physical path through IIS virtual directory. For instance,
http://localhost/CommonLibrary/WebGrid/V5_0_7200/Localization/default.xml

Sample codes in C#:

```
WebGrid1.LayoutSettings.TextSettings.UseWebResources = false;
```

There are several scenarios where *SmartWebResources* need to be disabled for localization:

- You want to use modified version of the language file.
- You want to use new language file that is not included in WebGrid package.

Tips: If you want to change only specific item of text for webpage-specific scenario, you can override the text item by adding the entry in the *TextItems* collection of a WebGrid instance. This enables WebGrid to use *SmartWebResources* for the selected language and use the overridden *TextItem* that you have specified.

Verbose Editing InfoText

WebGrid.NET Enterprise 5.0 comes with many enhancements related to editing feature. One of the most significant enhancements is the new *verbose information text in editing mode*. This new feature is useful to deliver intuitive editing experience for users as they can notice on important information related to a specific column in *status bar* area.

This new feature allows you to assign information *text* and *image* for each *WebGridColumn*. You can try to click on different cells in this sample and notice the verbose infotext in the status bar. When this feature is enabled, there are several automated behaviors:

- Column with EditType = NoEdit (read only) will have the status text and icon configured automatically.
- When *EditInfoImage* is provided, the image will appear in status bar.
- When a cell is being edited, the *Edit Image* will appear in status bar.
- When a cell is being edited or has been modified - and *EditInfoText* is not provided, the status text and image will be configured automatically.
- When the edit cursor is focused on cell which has been modified, the *Edit Image* indicator will appear in the status bar. This allow users to easily discover whether a cell has been modified or not.
- New user-friendly text and icon will be displayed after the successful add/edit/delete operation. For instance, the text *Record is updated successfully* will be displayed instead of *Ready* after successful update.

Automatic Filter Suggestion

New in version 5.0 is *Automatic Filter Suggestion* feature enabling users to easily perform data filtering based on the currently selected cell.

This feature automatically adds filter suggestions based on the data type and value of the selected cell. The benefits are:

- Allow users to quickly perform filtering without has to type in the filter bar.
- AllowFiltering/FilterBar can be turned off while allowing user to perform filtering on existing data.

To enable this feature, simply set *AutoFilterSuggestion* property of *LayoutSettings* object to True. This feature is best used along with *Filter Status Indicator* feature.

Filter Status Indicator

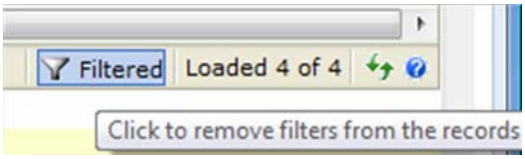
In conjunction with *Automatic Filter Suggestion* feature, Filter Status Indicator is a new user interface element which appear in the status bar. The filter indicator is useful to represent the current state of data when filter bar is not visible.

When this feature is enabled, it will automatically reflect the user's interaction in Filtering area. For instance, when you perform filtering from either context menu or filter bar, the filter indicator will be marked as active. Likewise, when all filters are removed, the indicator will be marked as disabled (inactive).

This new visual element enables you to:

- Quickly determines the current filter state of data.
- Easily toggle the filter state by clicking on the filter indicator.
- Visually enable filtering without the need to display filter bar.

To enable this feature, set *ShowFilterStatus* property of *LayoutSettings* object to True. The following image illustrates the filter status indicator user interface element:



Unbound Hierarchical

WebGrid.NET Enterprise 5.0 now allows you to configure Hierarchical child tables in Unbound mode. In previous versions, WebGrid only support databound mode to be used in Hierarchical feature. To simulate Unbound mode in previous versions, you usually create a temporary DataSet and the DataTables and map them through DataRelation.

In this new version, you no longer need temporary DataSet or workarounds in order to configure hierarchical Grid in Unbound mode. WebGrid 5.0 supports pure Unbound mode through server side object model.

Note that when the Grid is operating in Unbound mode, all data-bound features -- such as sorting, grouping, etc -- should not be enabled.

Loading PreviewRow On Demand

Oftentimes, preview row is used to display lengthy information such as Notes or Description. By default, WebGrid will render all preview row data at once in first load. This could introduce several performance issues due to large output size, and meanwhile user might not need to see all preview row's data at once.

In version 5.0, you can configure WebGrid to skip preview row rendering in first load to reduce initial page output size. In this scenario, the preview row *Expanded* state is automatically set to false (Collapse). The preview row data will be retrieved through AJAX callback when you clicked on the expand icon.

Displaying Aggregate Values in GroupHeader

Although built-in *Aggregate Functions* are already available in *WebGridColumn* in initial version, you need to enable *GroupTotal* feature to display the aggregated values. When enabled, a special total row will appear after the group header to display each calculated values of aggregate functions defined in specific WebGridColumn.

With the new feature to display basic aggregate functions in group header, you can display the calculated values without has to enable the GroupTotal feature. This new feature is useful when you only need to display a small number of calculated values and therefore reducing unnecessary clutter in the Grid's user interface.

The format string for new aggregate functions that you can apply in group header:

- sum(*Column Name*)
- avg(*Column Name*)
- min(*Column Name*)
- max(*Column Name*)

Multiple Values Support in WebValueList

WebGrid.NET Enterprise 5.0 includes enhancement in *WebValueList* feature to support multiple values data. In previous version, you can easily translate an ID field to user friendly field through *WebValueList* in WebGrid. However, you can only display one translated value. For instance, RoleID = *1* being translated to *Research Staff*.

In version 5.0, WebGrid makes it easy to translate multiple values data using the improved *WebValueList* feature.

To enable multiple values data lookup, you can set *EnableMultipleValues* property of the *WebValueList* object to *True*. You can also specify the separator character used to separate the values data in *MultipleValuesSeparator* property.

The following image shows the *Roles* column which values have been properly translated using *WebValueList*.

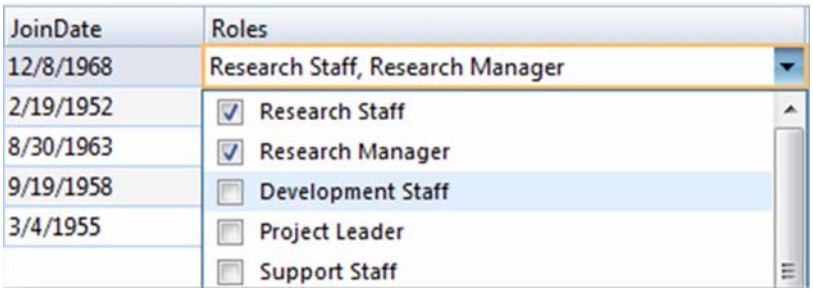
	LastName	JoinDate	Roles
	Davolio	12/8/1968	Research Staff, Research Manager
	Fuller	2/19/1952	Development Staff, Support Staff, Research Staff
	Leverling	8/30/1963	Research Manager, Support Manager
	Peacock	9/19/1958	Premium Support Staff, QA Staff
	Buchanan	3/4/1955	Research Manager, Development Staff, Support ...

Multiple Values Editing using WebCombo.NET 4.0 control

In conjunction with Multiple Values feature in the improved *WebValueList*, WebGrid.NET Enterprise® 5.0 also includes the ability to edit the multiple values data that translated using the *WebValueList*.

This powerful feature implements deep integration between Intersoft's *WebGrid* and *WebCombo* control. With *Multiple Selection* feature available in WebCombo.NET version 3.0+, the WebGrid control seamlessly connect to WebCombo control in order to edit and return multiple values data of the selected cell.

In addition to multiple values editing, WebGrid.NET Enterprise 5.0 supports deeper integration with WebCombo.NET 4.0 control in other areas such as in user interface and editing environment. For instance, when both WebCombo and WebGrid control are using Default Style, the WebCombo control's appearance will be adjusted automatically to fit into the editing cell. This automated behavior ensures rich editing experience with consistent look.



FAQ: Does WebCombo.NET 4.0 control included in WebGrid.NET Enterprise 5.0?

No, WebCombo.NET 4.0 license is not included with WebGrid.NET Enterprise 5.0 purchase. WebCombo.NET 4.0 is a standalone component and is sold separately. However if you owned a valid Subscription license, WebCombo.NET 4.0 should be available to you without additional costs.

Automatic Column Sorting using WebValueList

Since the introduction of WebGrid.NET Enterprise 3.1, *WebValueList* has been one of the most popular feature in WebGrid control. With WebValueList, you can easily translate *ID to Text Lookup* by using the datasource populated in the WebValueList object. For instance, the Supplier Name in

the WebGrid above is actually an ID column which is translated to *Company Name* field of *Suppliers* table through WebValueList.

In version 5.0, WebValueList feature has been significantly enhanced to overcome several limitations introduced in previous versions of WebGrid. The column sorting and grouping can now take advantage of the same datasource that populated in the WebValueList. With this enhanced WebValueList, you no longer have to manually retrieve the text field of the value column in the WebGrid's main datasource.

Sample codes (ASPX):

```
<ISWebGrid:WebGrid ID="WebGrid1" runat="server" Height="300px"
        UseDefaultStyle="True" Width="100%" DataMember="Products"
        DataSourceID="ISDataSource1">
         <RootTable DataMember="Products" UseValueListForSorting="Yes">
             <Columns>
             ...
                <ISWebGrid:WebGridColumn ...>
                     <ValueList DataMember="Suppliers"
                         DataTextField="CompanyName"
                         DataValueField="SupplierID">
                     </ValueList>
                 </ISWebGrid:WebGridColumn>
             ...
             </Columns>
         </RootTable>
         <LayoutSettings AllowSorting="Yes" AllowGrouping="Yes">
         </LayoutSettings>
</ISWebGrid:WebGrid>
```

Built-in Cell Validation through new Required Input

WebGrid.NET Enterprise 5.0 now provides built-in feature for *Required Input* during data editing and makes data validation a snap.

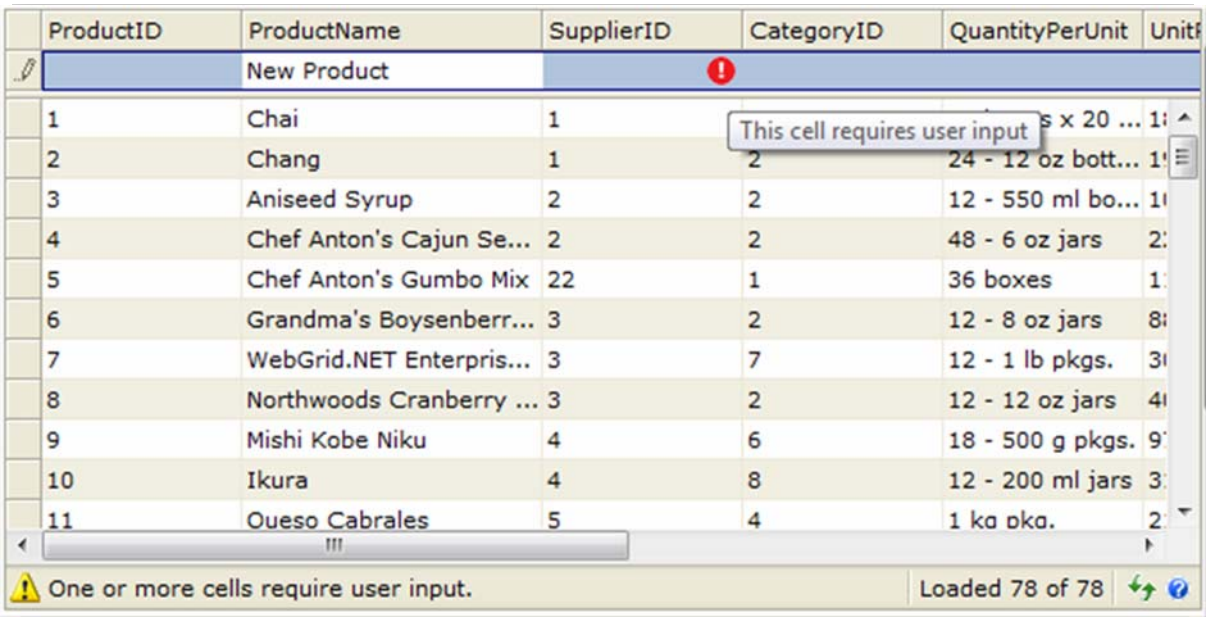
When the cell of the specified column is empty, the updating will be blocked until all required input are filled. To enable user input, simply set *InputRequired* property to true at *WebGridColumn* level. You can also modify the error message in the *InputRequiredErrorText* property.

Things to experiment:

- Click on one of the existing record. Double click on *SupplierID* and clear the value.
- Click on another row to perform updating. Notice that the updating is blocked.
- Also notice that an error indicator will appear in SupplierID. Try to hover the error indicator to see the required error text.
- Now try to clear the *UnitPrice* and try to perform updating again. Notice that the required error text is customized.
- Notice the warning message text in the status bar.

With this built-in feature, data validation for requiring user input can be achieved easily without has to write single line of codes.

The following image illustrates the required input feature.



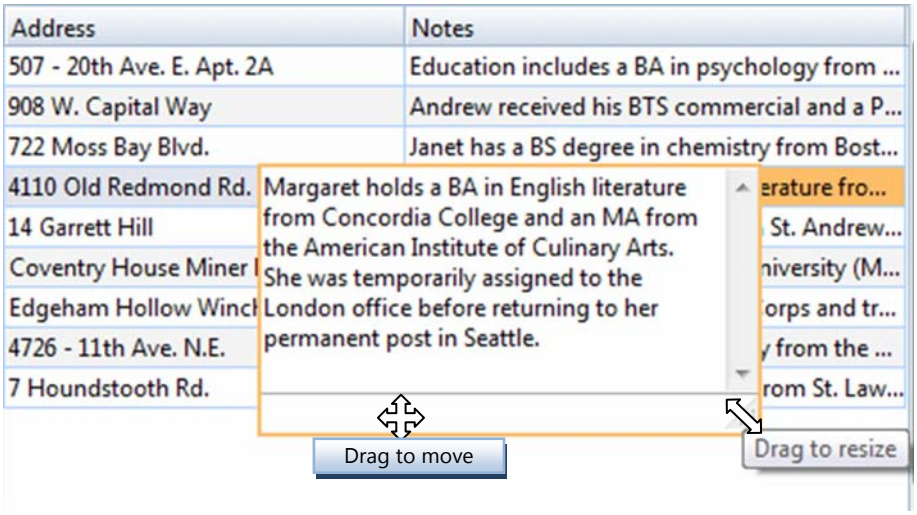
New Edit Type "Resizable TextBox"

WebGrid.NET Enterprise has introduced rich editing experience since the initial version, such as ability to perform data editing directly inside the Grid. Additionally, WebGrid also includes several rich editing types that you can use in the WebGrid.

In version 5.0, WebGrid.NET Enterprise® redefines the editing experience by introducing a new edit type, *resizable text box edit control*. The resizable text box makes it easy and convenience for user while editing cell containing larger amount of text. With this new feature, user can easily resize the text box by dragging the *resize handler* at the bottom right position.

When a cell has been modified, you can press enter to accept the modified value as in normal text box. Note that you can also move the textbox control by dragging the empty space in the bottom bar. Combine it with resizing feature to fit your need for your editing convenience.

The following image illustrates the *Resizable TextBox* in *Notes* column.



Improved Data Caching (for traditional data binding)

The data caching feature in version 5.0 has been significantly improved to avoid memory leaks and lower usage of allocated resources when the *DataCacheStorage* is set to *PageCache* (default).

Note that the built-in data caching is disabled by default when the WebGrid is bound to *datasource control*. The data caching is provided for the usage when the WebGrid is bound to datasource object (i.e. through *InitializeDataSource* event). When bound to *datasource control*, the data caching is handled at datasource control level (through *EnableCaching* property) and thus no longer need to be handled by WebGrid.

For more information about datasource control support, see Declarative DataSource Control

Another new feature related to built-in data caching is the ability to store the cache to physical disk. WebGrid now adds a new option in the *DataCacheStorage* called "FileServer". When you choose this option, the data cache will be stored to the path defined in **CacheServerConnection** property. This new option allows better scalability and reliability to your web application, as the data caches are no longer stored in ASP.NET's worker process or Session.

For more information about FileServer feature and its benefits, see *New Cache-To-Disk option for ViewStateStorage™* [TODO: Link to WebUIFramework2007 Documentation]

Templated Cell

Another interesting new feature in WebGrid.NET Enterprise 5.0 is its ability to add ASP.NET server controls to the WebGridCell using the new *Template Column Type*. When using the *Template Column Type*, you can put any server side controls inside the *CellTemplate* property of the *WebGridColumn*.

This capability brings you greater flexibility and control over the data visualization and manipulation that suitable to your application needs. Given the current trend and ability of AJAX in delivering

changes from server side control without page postback, this new feature enables developers to achieve broader scenario using professional server-side programming.

You can use *Eval* or *Bind* method for declarative data binding in the child controls of the *CellTemplate*.

For instance,

```
<ISWebGrid:WebGridColumn Caption="ContactName" ColumnType="Template"
      DataMember="ContactName" Name="ContactName" Width="200px">
      <CellTemplate>
            <asp:Label ID="Label1" runat="server" Text='<%#
            Bind("ContactName") %>' ForeColor="Red" Font-Italic="true"
            />
      </CellTemplate>
</ISWebGrid:WebGridColumn>
```

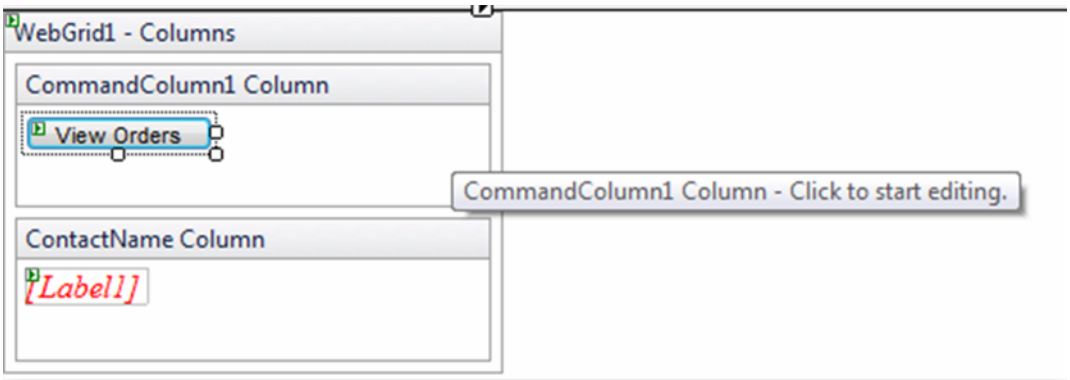
In order to allow grouping, sorting, and editing on Templated Column, the *DataMember* of the *WebGridColumn* need to be specified. This enables WebGrid to use native value of represented row item for the data operation purpose.

Editing is not affected with Templated Column. This means you can still specify the *EditType* of the Column, while it is displayed using *Template ColumnType*.

To edit the templated cell in Visual Studio 2005 design-time, the template editing options will automatically appear when it detected *Templated ColumnType* in the WebGrid. The template editing can be performed by right clicking on the WebGrid instance, then point to Edit Template menu, and finally click on *Columns* menu item.

Once you have entered template editing mode, you can drag and drop server side controls from your toolbox into the provided region surface. You can then customize the properties of the server controls or wire events to perform logic handing.

The following screenshot shows the *template editing mode* in Visual Studio 2005.

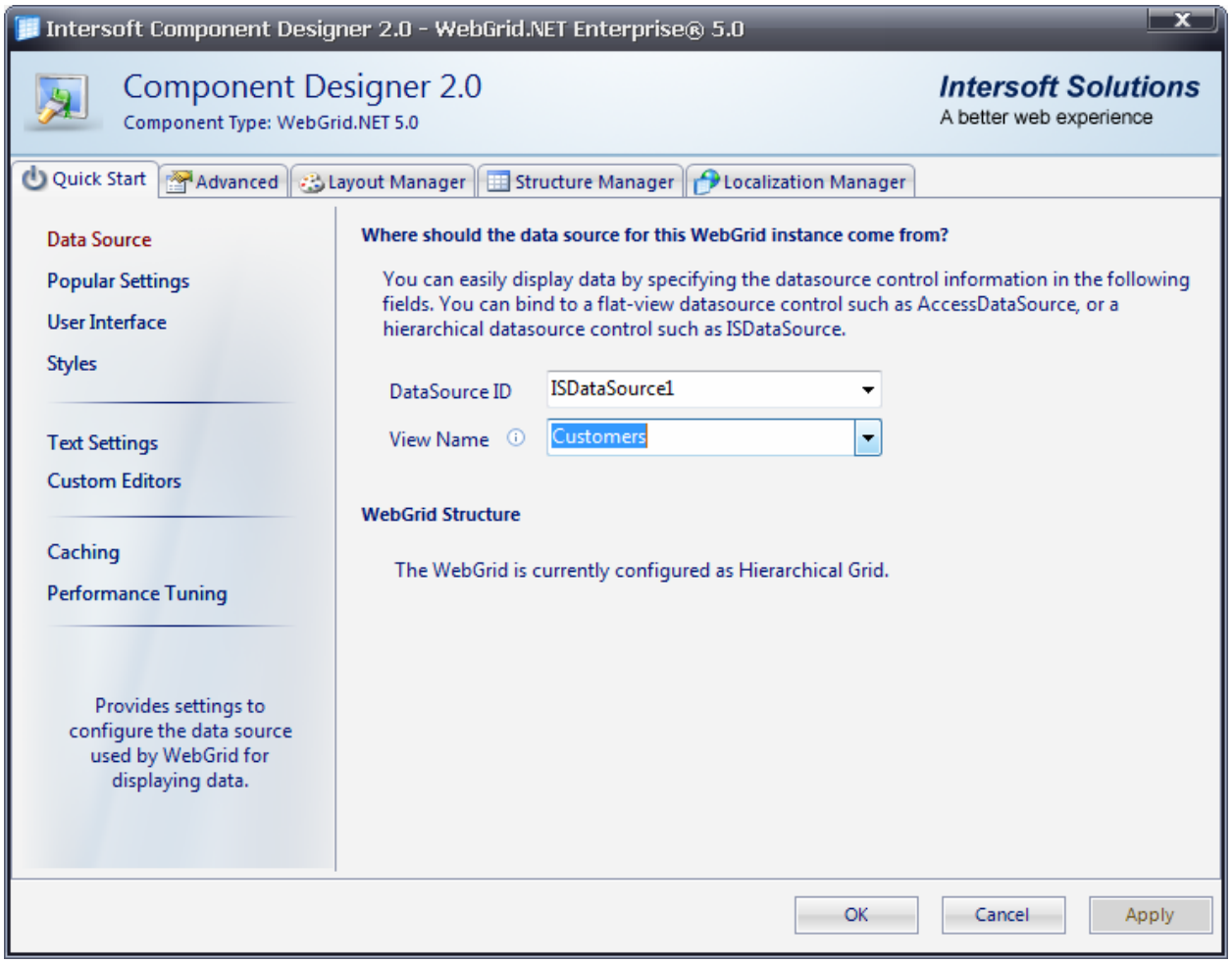


WebGrid.NET Component Designer 2.0

WebGrid.NET Enterprise 5.0 takes web development to a new level by introducing rich component designer to help developers easily configure the powerful and advanced settings available in WebGrid.

Based on Intersoft's state-of-the-art Component Designer 2.0 architecture, *WebGrid.NET Enterprise 5.0 Designer* is a highly advanced component editor delivering unmatched ease-of-use and streamlined user interface.

WebGrid.NET Enterprise® 5.0 Designer comes with 5 main tab-style interfaces such as shown in the following screenshot.



The main user interface of WebGrid Designer in version 5.0

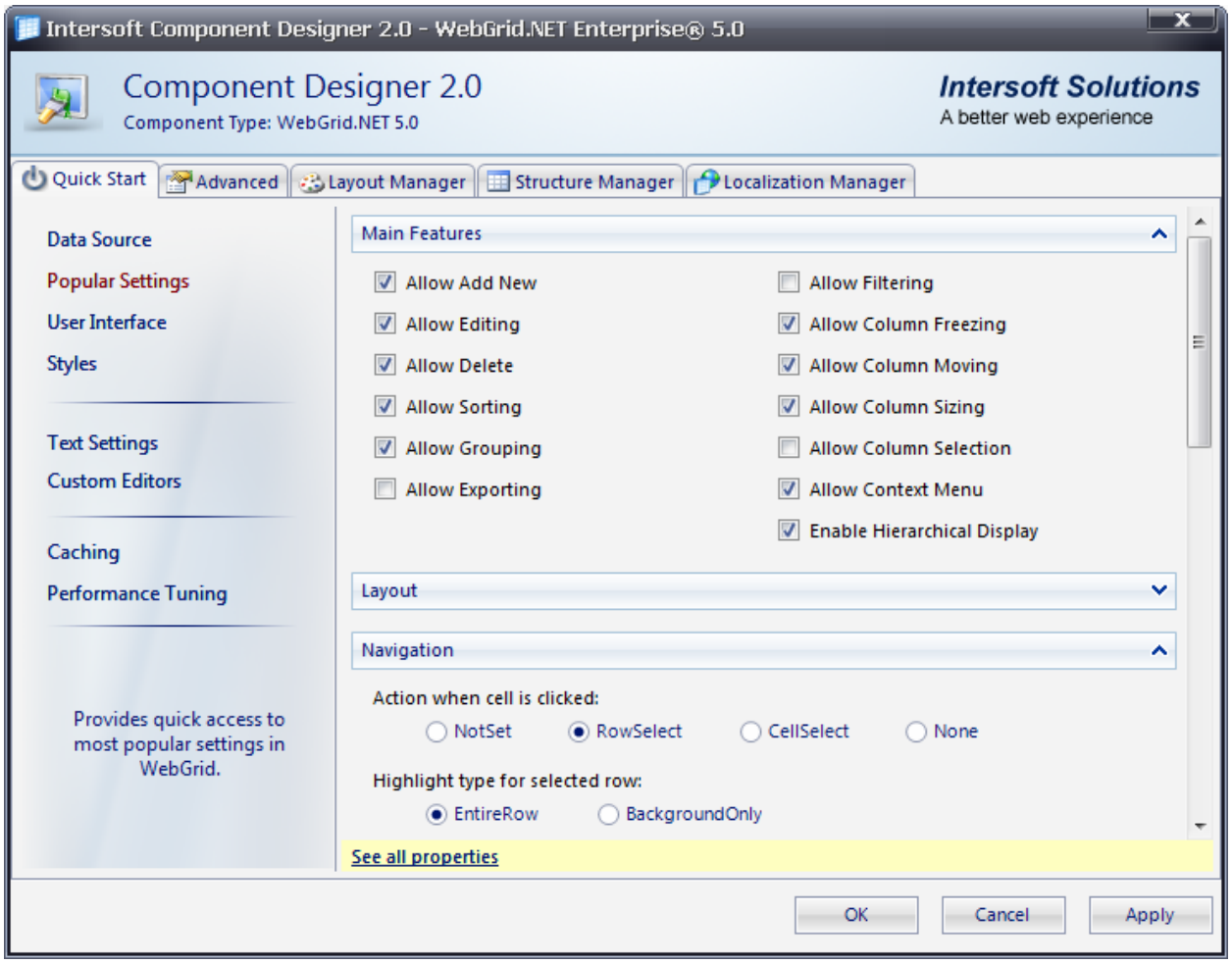
The following sub topics discuss the functionality of each tab.

Quick Start

The *quick start* page is the main access point of most popular settings to help you quickly getting started with WebGrid. By default, the quick start page is displayed automatically when launched from design surface.

The quick start page consists of 8 sub pages. The *Data Source* sub page is automatically displayed when the WebGrid doesn't have binding configuration yet. The *Popular Settings* sub page will be displayed when the WebGrid already has a valid binding configuration. This allows you to concentrate on WebGrid's customization based on its current state.

The following image shows the screenshot of *Popular Settings* in *Quick Start* page.



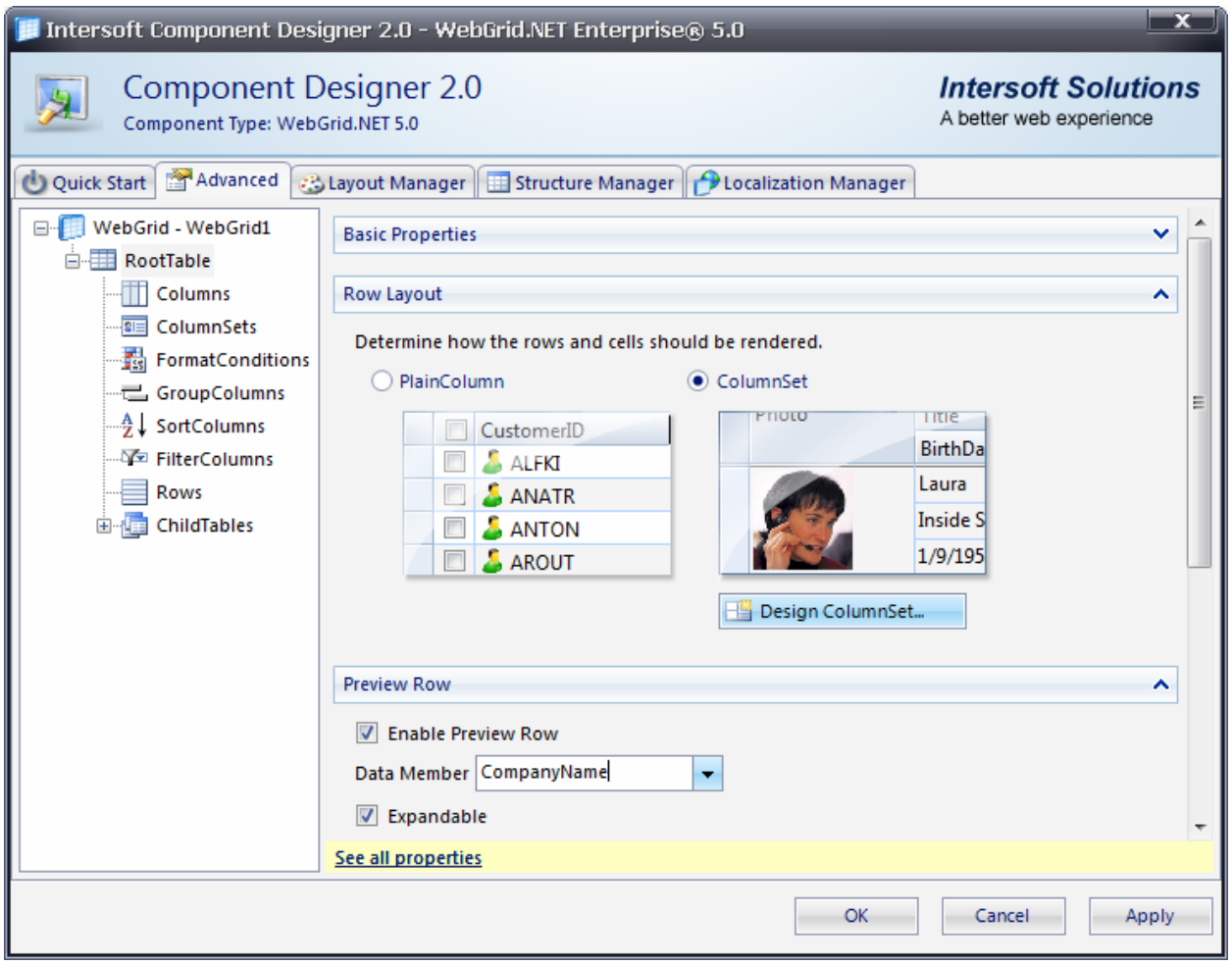
The new expandable and collapsible user interface allows you to quickly locate a feature. Most settings are presented in check box and radio selection interface which designed to help you instantly enable or disable, or modify a feature with one mouse click.

Advanced

The *Advanced* page gives you complete control over the WebGrid configuration and its tables structure. While most main features can be easily enabled in Quick Start page, some advanced features such as column set display, self referencing, or other table-level settings can only be configured in Advanced page.

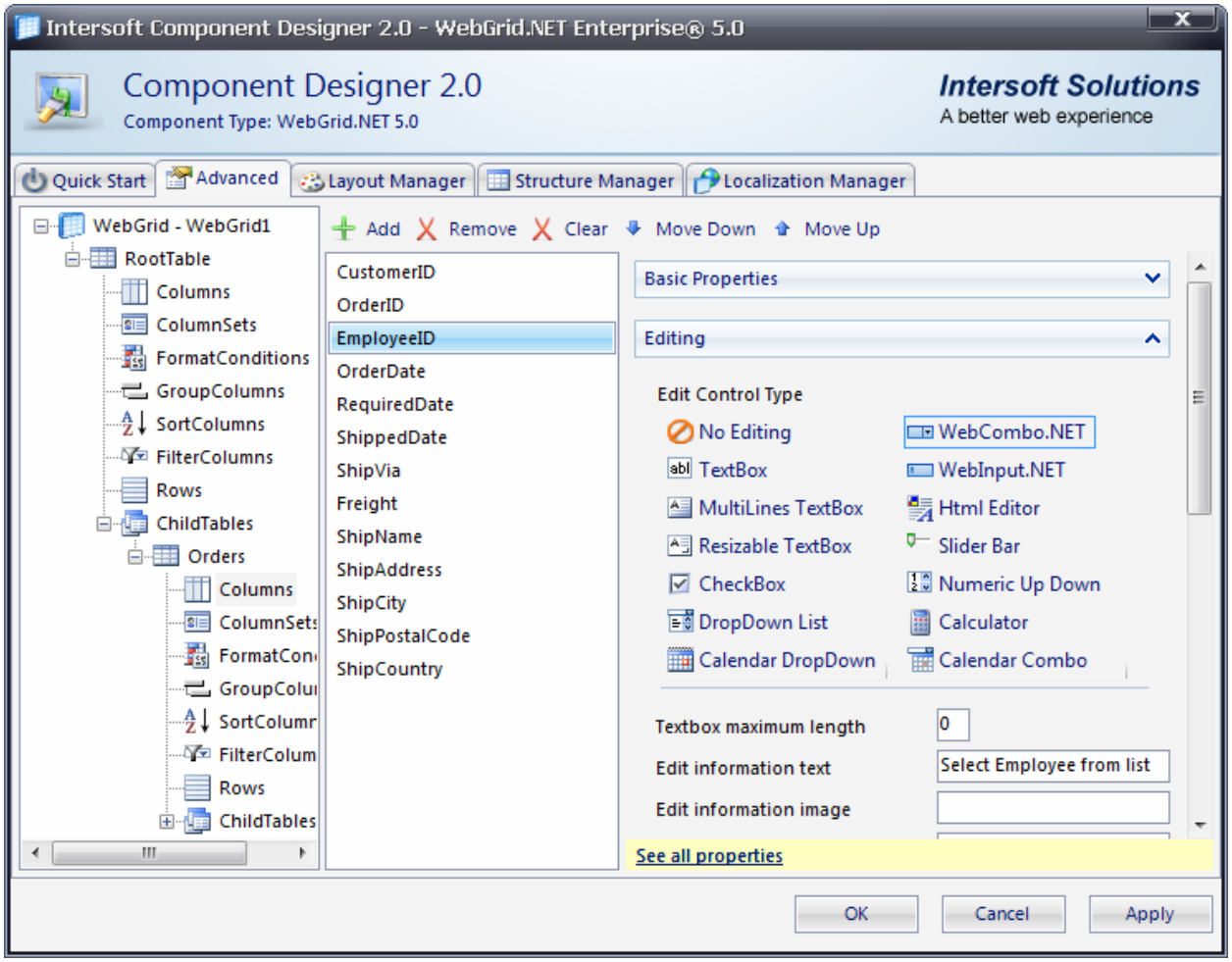
Combining the state-of-the-art *Component Designer* with streamlined user interface concept in WebGrid.NET Enterprise 5.0, the *Advanced* page also displays rich designer for several main objects such as *WebGridTable* and *WebGridColumn*. This results in sophisticated designer experience and dramatically increasing developer's productivity.

The following image shows the screenshot of Advanced page, while focused at a *WebGridTable*.



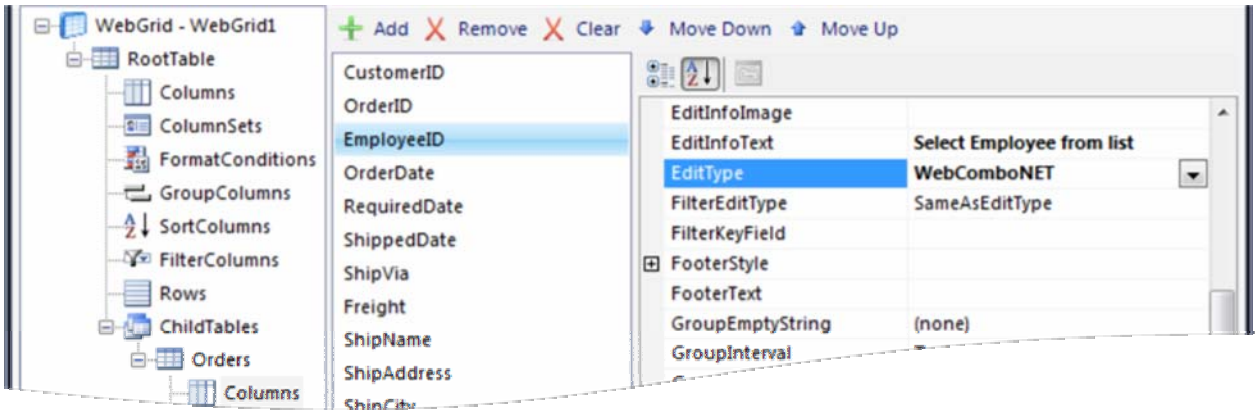
With rich designer, you can now easily modify advanced configuration. For instance, you can launch ColumnSet Wizard by clicking on *Design ColumnSet* button in this screen, instead of having to set related settings in several locations.

The following image shows the screenshot of Advanced page, while focused at a *WebGridColumn*.



Thanks to the rock-solid *Component Designer* architecture that made rich designer experience possible, you can now instantly select an editing control for the selected column by simply clicking on the visual clue. No more tedious steps and lengthy procedure to use a custom editor such as Calculator or Html Editor.

The following image shows the screenshot of Advanced page, when *See all properties* link is clicked.



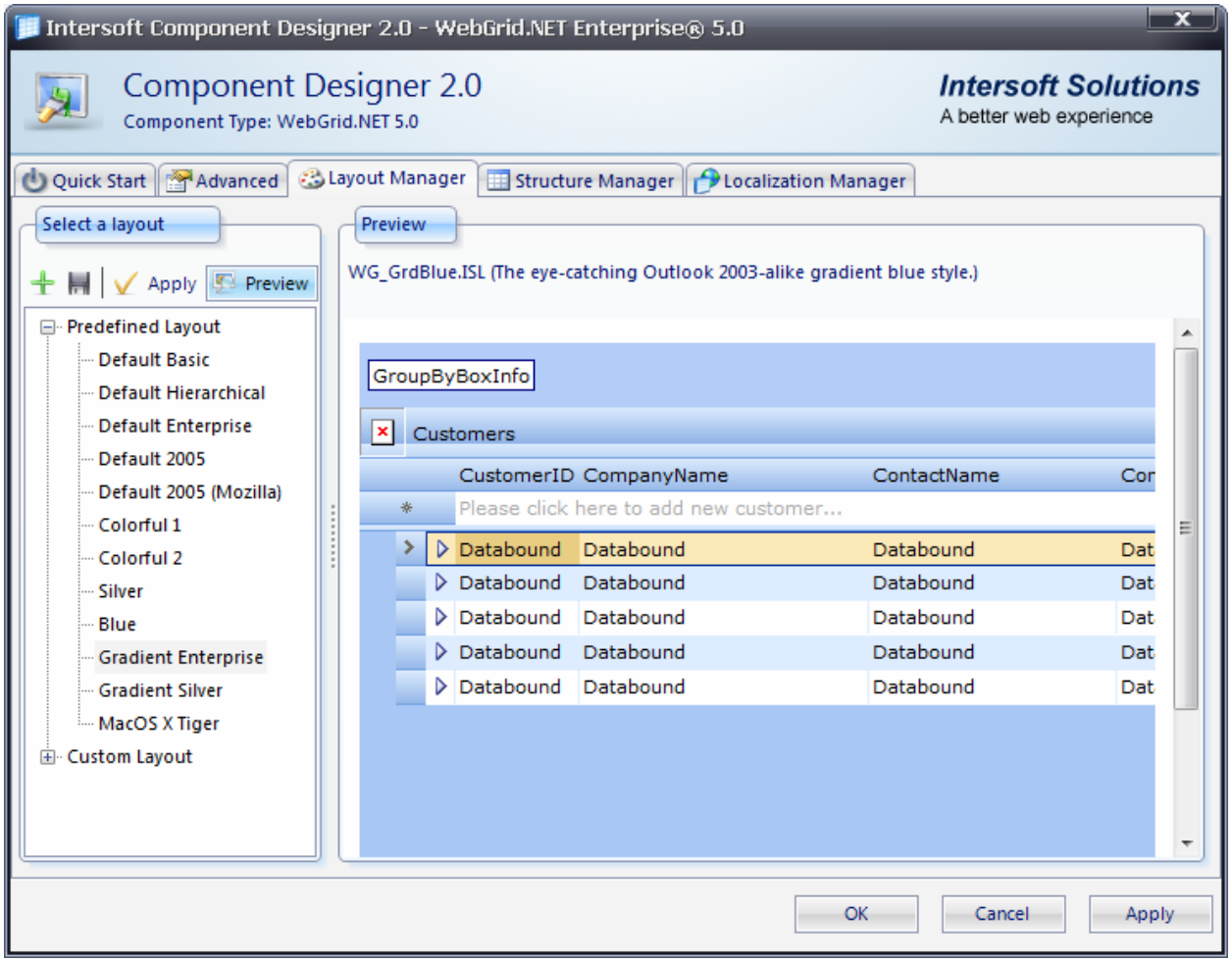
For advanced developer who preferred to browse all properties, here is the mode for you.

Layout Manager

The Layout Manager feature has been introduced since previous version of WebGrid.NET Enterprise. The feature was previously implemented in WebUI.NET Framework as part of Component Designer 1.0. Layout Manager is a powerful feature to let you easily apply from a predefined style as well as save your own style and reuse it later.

The *Layout Manager* in second generation of Component Designer has been much improved in many areas. It now has cleaner look and uncluttered user interface to bring you larger space for the preview.

The following screenshot shows the new Layout Manager in Component Designer 2.0.



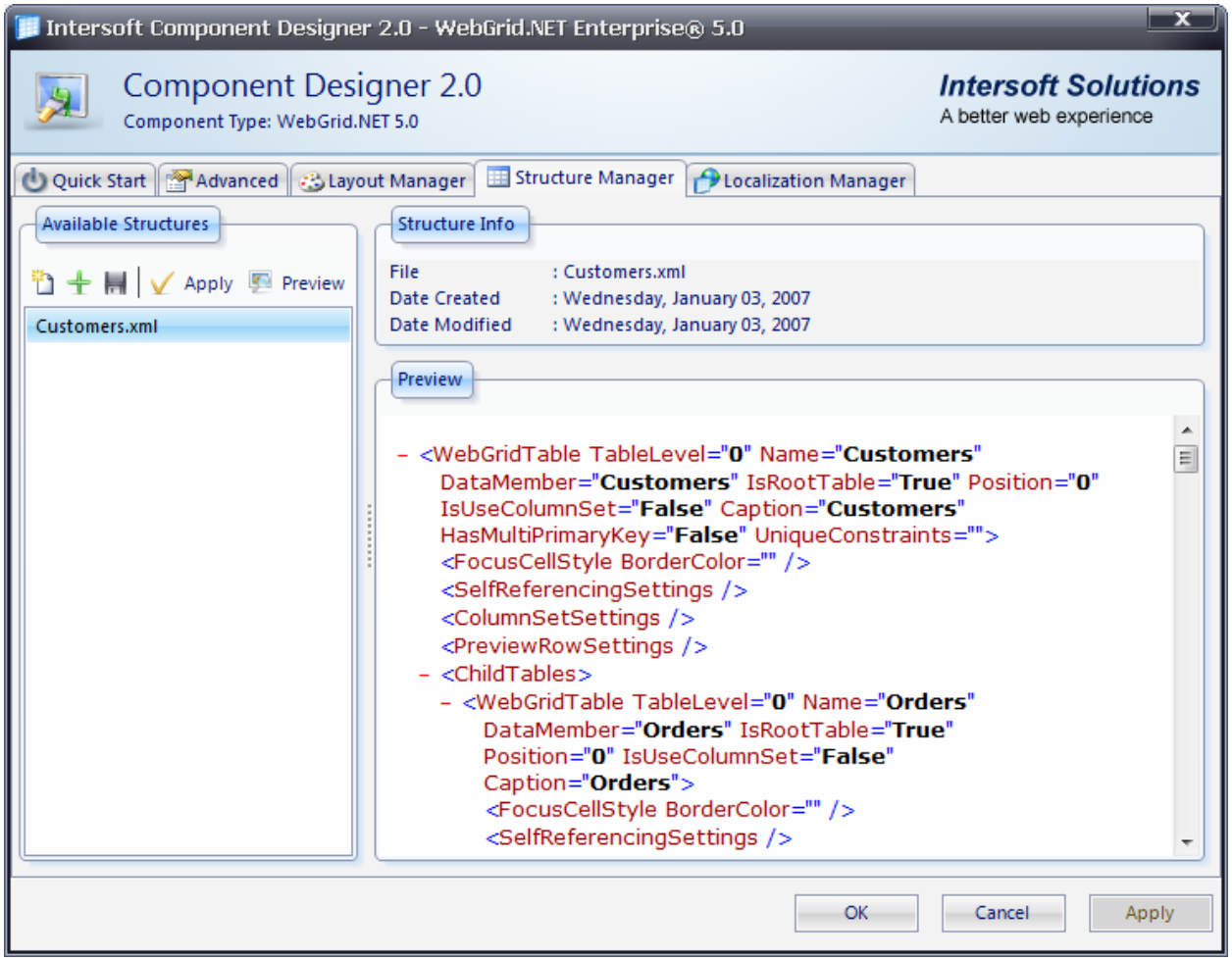
You can easily preview a style before decide to apply it by simply clicking on Preview command. The preview is performed in real time and is showing the exact visual appearances as in runtime.

Structure Manager

Structure Manager is a new feature in WebGrid.NET Enterprise 5.0. This feature enables you to save the structure of an existing WebGrid in design-time. The structure includes table configuration such as columns, grouped columns, filtered columns as well as child tables.

With this feature, you can easily reuse a WebGrid's structure by applying the saved configuration in Structure Manager.

The following image demonstrates *Structure Manager* page in action.



Structure Manager allows you to preview an existing structure before you choose to apply it. This feature increases productivity by enabling developers to reuse a complex structure in multiple instances or projects.

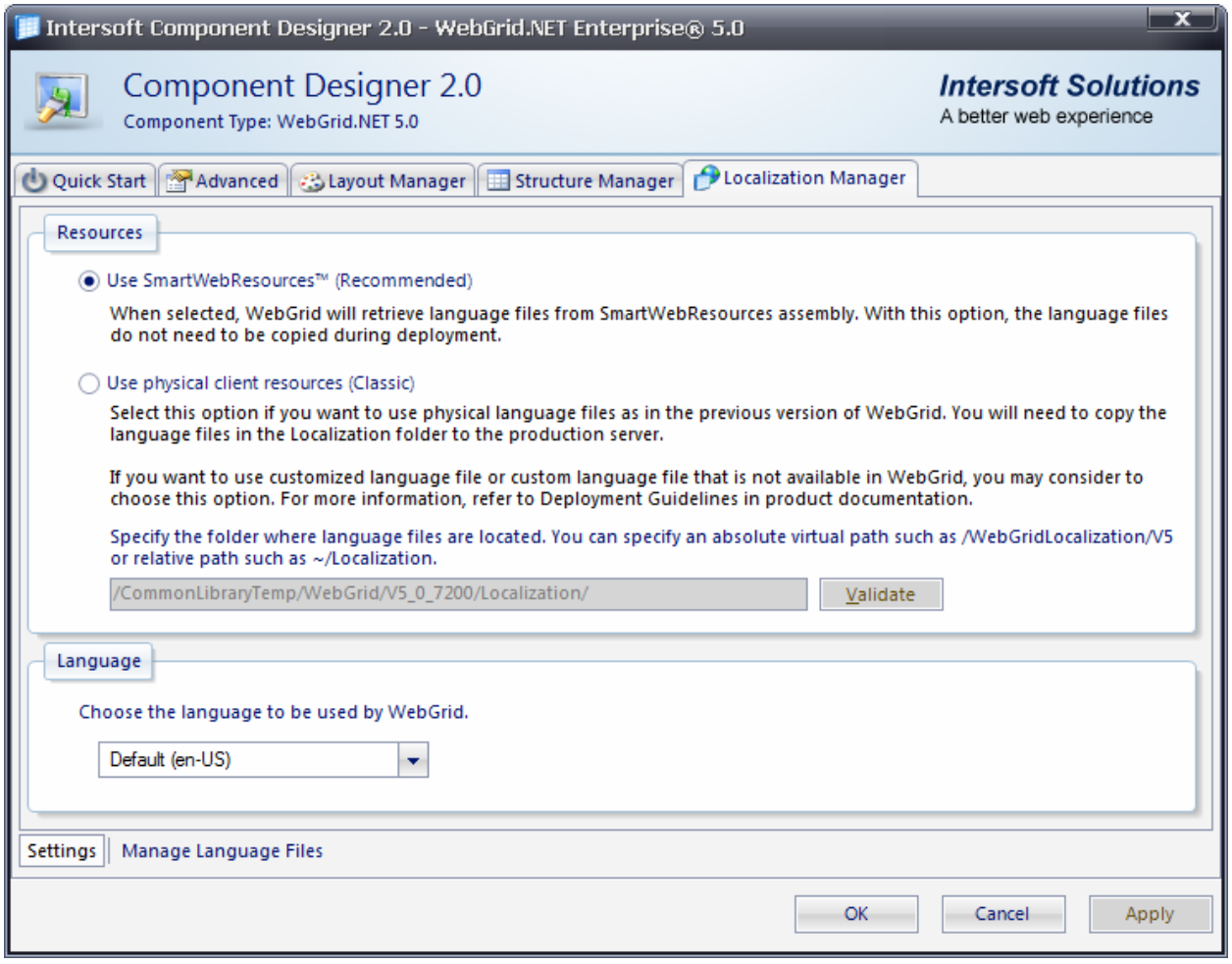
Localization Manager

Localization Manager is a feature to help you configuring the language settings in WebGrid. In this new version, *Localization Manager* has been much improved with the adoption of clean and uncluttered user interface which results in better development experience and easier to use.

The improved Localization Manager also incorporates the new SmartWebResources™ feature in its user interface so that you can effortlessly configure resources and related language settings in one streamlined user interface.

Note that when the language resources are retrieved from SmartWebResources™, the WebGrid requires the main SmartWebResources™ setting to be enabled. This is the recommended setting in WebGrid.NET Enterprise 5.0 as well as other 2007 platform products. To learn more about SmartWebResources™ feature, see Hassle-free deployment through SmartWebResources™ technology

The following image shows the main user interface of Localization Manager.

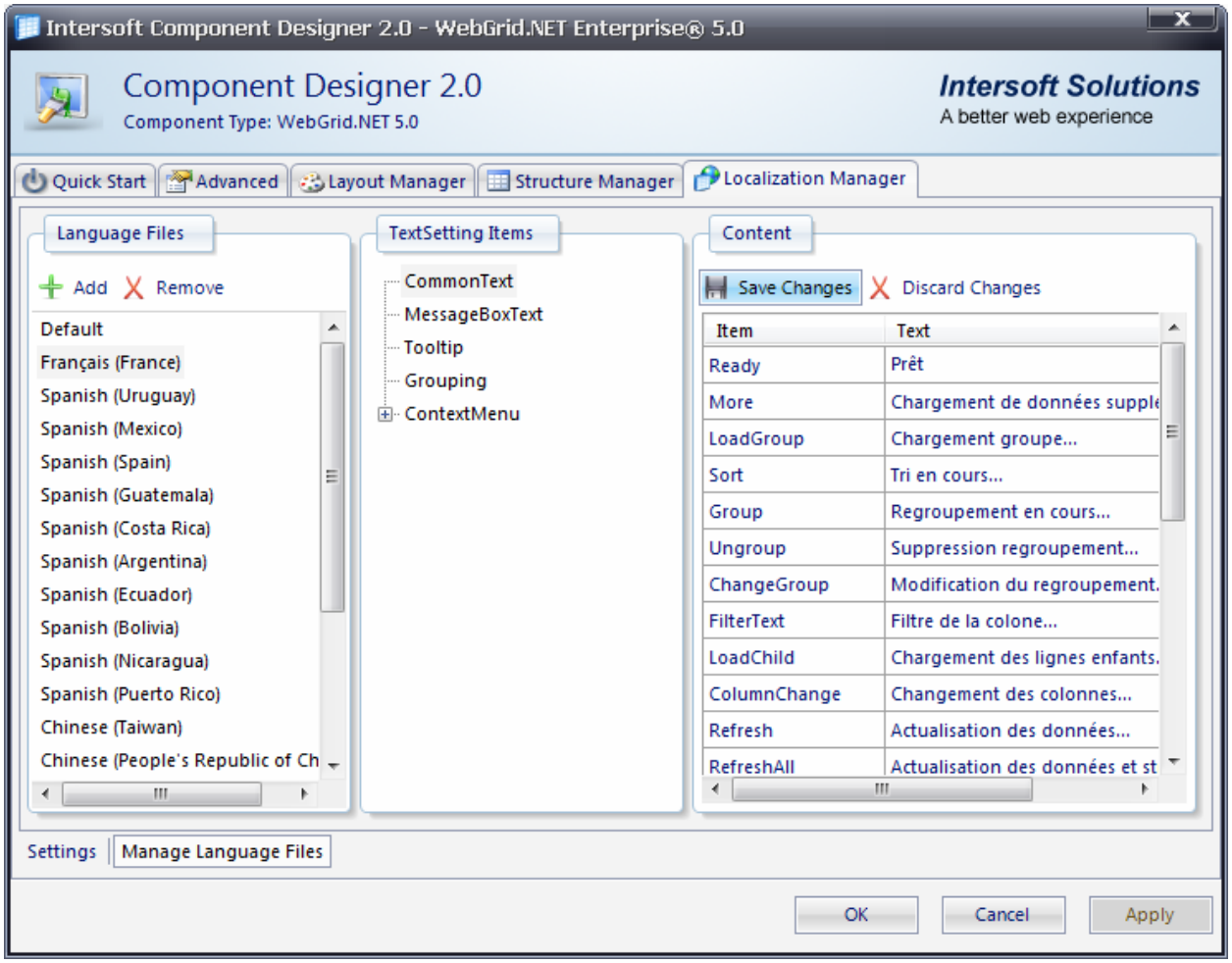


The clean, streamlined user interface to manage resources and language-related settings in WebGrid.NET Enterprise 5.0

If you would like to use modified language file, or your own language file which is not shipped along with WebGrid, you need to use *Physical Client Resources* mode so that your language file can be retrieved through classic web client url.

When you used *Physical Client Resources* mode, you can manage the language list from the user interface. You can also customize the text entry in the *Manage Language Files* sub page.

The following image shows the *Localization Manager* with *Manage Language Files* sub page activated.



The *Manage Language Files* feature lets you add or remove a language file, or customize the existing language file with your own customized text entry.

Enhancements

In addition to dozens of exciting new features introduced in version 5.0, WebGrid.NET Enterprise® also includes enhancements to performance, server side and client side object model and several important areas such as discussed in following sub topics.

Server-side object model

Most significantly enhanced server-side object model in version 5.0 are related to new classes' constructors and overloads for commonly used classes. For instance, you can now create several cells and rows in one line of code, and add them at once using *AddRange* method of the collection.

Sample codes in C#:

```
WebGridRow[] rootRows = new WebGridRow[6];

for (int i = 0; i < 6; i++)
    rootRows[i] = WebGrid1.RootTable.CreateRow();

rootRows[0].Cells.SetItemData(new string[] { "172-32-1176", "White",
   "Johnson" });
rootRows[1].Cells.SetItemData(new string[] { "213-46-8915", "Green",
   "Marjorie" });
rootRows[2].Cells.SetItemData(new string[] { "238-95-7766", "Carson",
   "Cheryl" });
rootRows[3].Cells.SetItemData(new string[] { "267-41-2394",
   "O'Leary", "Michael" });
rootRows[4].Cells.SetItemData(new string[] { "274-80-9391",
   "Straight", "Dean" });
rootRows[5].Cells.SetItemData(new string[] { "341-22-1782", "Smith",
   "Meander" });

WebGrid1.RootTable.Rows.AddRange(rootRows);
```

In above sample, the *CreateRow* is a new method of *WebGridTable* class which creates and initializes *WebGridCell* according to the columns and structures available in *WebGridTable*. The *SetItemData* method allows you to easily set an array of string which will be mapped to the cells in *WebGridCellCollection* respectively. The *AddRange* method finally perform the addition of the *WebGridRow* arrays to the *WebGridTable*.

In previous version, the same sample could take 2-3 times more codes compared to above codes. The enhanced server side object model in version 5.0 enables developers to access and manipulate the WebGrid control in less efforts and codes.

Client-side object model

- New methods and objects have been added to client-side object model in version 5.0 to reflect new API that added to server side. The following is the details of newly added objects and methods:

Class Name	Member	New	Type
WebGrid	FreezePane	✔	Method
	UnfreezePane	✔	Method
	GetLastFrozenColumn	✔	Method
	SetStatusIcon(*mode, isCustomImage*)	✔	Method
	ClearSelection	✔	Method
WebGridCell	GetAttribute(*attrName*)	✔	Method
	IsEditable	✔	Method
WebGridColumn	GetWidth	✔	Method
	GetNewRowEditType	✔	Method
WebGridTable	IsFilterDisabled	✔	Method
	GotoFilterRow	✔	Method
	GotoNewRow	✔	Method
	SetFilterStatus(*isEnabled*)	✔	Method
WebGridFreezePaneSettings	All properties are equal to server side	✔	Class
WebGridRow	GetSelfRefChildRows()	✔	Method

- API performance is now faster up to 80% compared to previous version. Check out *Sample: High Performance Client-side API*

Client events and user interaction

- Added *OnScroll* event to client side event.
- Added *OnAfterInitialize* event to client side event.
- Ability to navigate through different groups using the *Tab* key.

Exporting

- Added *OutputType* property to *ReportInfo*. The temporary report file name format is now set to use *Guid* by default. In previous version, the default value is *ExecutionTime*.
- Exporting is now working properly without default printer installed. The error message like “Printer not Installed” or “PaperKind does not supported” has been completely eliminated.
- Reduced memory usage and fixed leaks.
- Core engine has been rewritten for better performance.

- Added *CacheToDisk* property to *ReportInfo*. This feature allows the exporting to write temporary cache to disk instead of using memory.
- Added *CacheToDiskLocation* property to *ReportInfo*. You can customize the folder to store the cache. By default, you don't have to set this property.
- Improved exporting menu. The export types are now represented with icons.
- Added exporting command to row/cell context menu. This allows you to easily perform exporting from your current mouse position in the Grid.

User Interface

- Unhandled server exception's error box can now be skipped. By setting *DisplayDetailsOnUnhandledError* property to *False*, the error box will not be displayed immediately. Instead, the error indicator will appear in the status bar. You can then click on the status bar to display the error detail message box.
- Pixel-perfect rendering for Mozilla-based browsers. In version 5.0, the box dimensions for cell, row and other user interface has the equal layout as displayed in Internet Explorer.
- New *RowHighlightType* property. The *BackgroundOnly* option allows WebGrid to redraw only the background color for selected row.
- To learn more new features related to User Interface, see New User Interface, New Styles

Core engine on output optimization

- Reduced output size up to 60%.
- When culture is set to default (en-US), the xml data for culture info is no longer generated in the webpage.
- When default style is enabled, in-line styles are no longer generated in the webpage. Instead, all WebGrid instances in the same webpage will share the same external stylesheet which significantly reducing output size.

Hierarchical

- Supports child row preloading.
- Supports true unbound mode through server-side object model.

Editing

- *PromptBeforeDelete* property is now affected when user tries to delete a row from context menu. In previous version, *PromptBeforeDelete* only works when user press Delete key to delete a row.
- *HtmlEditor* Editing Control now supports *SmartWebResources*. This allows you to use HtmlEditor without has to worry about client images, scripts and the popups required by the editing control.
- *HtmlEditor* Editing Control now supports Mozilla-based browsers. [Shifted to Beta2]
- *Calculator* Editing Control now supports Mozilla-based browsers.

Filtering

- Filter types now displayed based on column's data type. Unrelated filters will be automatically removed from the list. This enhancement is automatic and does not need further configuration.
- Special date range filters. When autofiltersuggestion is set to true, user can choice wider choice of date range filters in the column's or cell's context menu. The special date range filters are like Today, Tomorrow, Yesterday, Next Week, Next Month and so on.
- Added "Contains" and "Does Not Contain" as new filter types for string data type.

Context Menu

- Runtime engine is now based on Intersoft's *Hybrid Menu Runtime System*™. The Hybrid Menu Runtime System is the core menu engine used in *WebMenu* component, as part of *WebDesktop.NET*.
- As the result of new runtime engine, the display performance is significantly faster and has better stability. The context menu now has pixel-perfect layout between Internet Explorer and Mozilla-based browsers.
- New context menu styles and appearance with new icon-sets.
- Comprehensive menu commands on column and row level context menu.
- Commands are improved in most areas of feature that depend on Context Menu, such as in *Exporting* feature and so on.

Self Reference

- Ability to select the child row of a specified parent row in Self Reference Grid from server side.

  Sample C# codes:

```
protected void WebGrid1_InitializeRow(object sender,
ISNet.WebUI.WebGrid.RowEventArgs e)
    {
        if (e.Row.Cells.GetNamedItem("FirstName").Text ==
"Andrew")
        {
            e.Row.ExpandSelfRefRow();
        }
        else if (e.Row.Cells.GetNamedItem("FirstName").Text ==
"Janet")
        {
            e.Row.Selected = true;
            WebGrid1.SetFocus();
        }
    }
```

  To learn more, refer to *WebGrid.NET Enterprise 5.0 Tutorial* > *SelectSelfRefChildRow.aspx*

- Ability to get the self reference child rows of a specified parent row in client side.

Sample codes in JavaScript:

```
function GetChildRows()
  {
      var grid = ISGetObject("WebGrid1");
      var selObj = grid.GetSelectedObject();

      if (selObj != null)
      {
          var row = selObj.GetRowObject();
          var srChildRows = row.GetSelfRefChildRows();

          if (srChildRows.length > 0)
          {
              var firstChild = srChildRows[0];

              alert("Total child rows: " + srChildRows.length +
                 "\n" +
                  "First child row's FirstName Text: " +
                 firstChild.GetCell("FirstName").Text);
          }
          else
              alert("There are no child rows for this parent.");
      }
  }
```

To learn more, refer to *WebGrid.NET Enterprise 5.0 Tutorial* > *GetSelfRefChildRow.aspx*