PNMsoft Knowledge Base

Sequence User Guides

UX Studio Defining Controls

March 2014
Product Version 7.0 and above

PNMSOFT
Business Processes in Motion™

PNMsoft UK 38 Clarendon Road Watford Hertfordshire WD17 1JJ

Tel: +44(0)192 381 3420 • Email: info@pnmsoft.com • Website: www.pnmsoft.com

Microsoft Partner
Gold Application Development

PNMSOFT
Business Processes in Motion™

TABLE OF CONTENTS

Advanced Form Controls and Operations 1

Adding a Grid to a Form 1

Creating a Custom Grid Add/Edit Record Form 10

Defining Filtered Grids 11

Grid Features 11

Defining an Advanced Combo Box 13

Defining Filtered (Interconnected) Combo Boxes 14

Combo Box Caching Items 16

Form Table Editing 17

Managing Multiple Views 18

Loading Form Sub Views on Demand 20

Passing Parameters from Main Form to Sub View 21

Save Command Options 24

Implementing Validations 26

Validation Groups 27

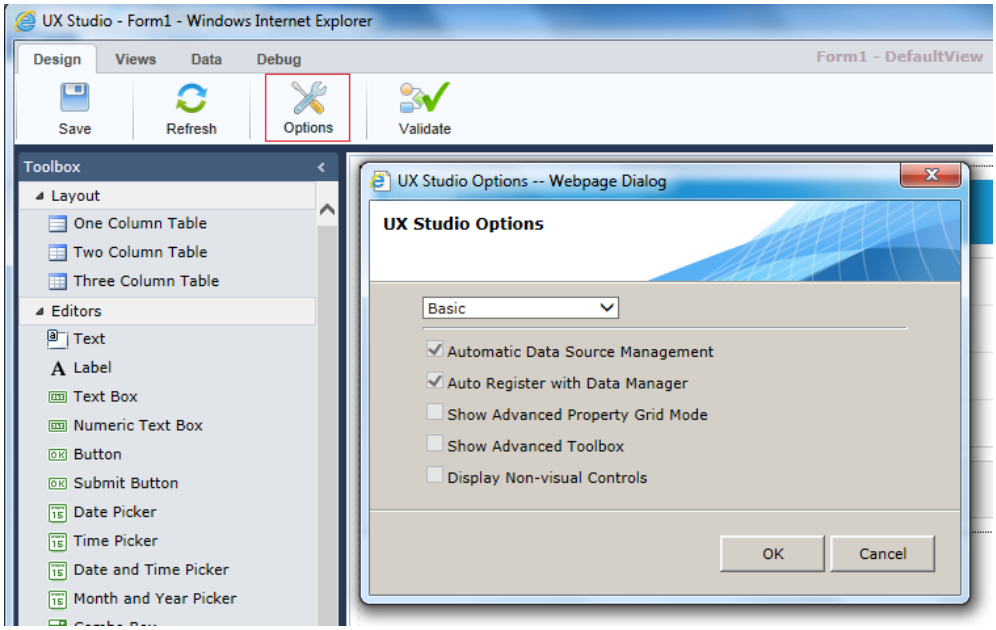
Example of Compare Validator 27

Required Field Validator 28

Using Ajax Update Panels 28

Advanced Form Controls and Operations

Note: The guide assumes you are using the Options > Basic view. Some features in the guide require the Advanced view, and are noted as such in their section.



Basic View

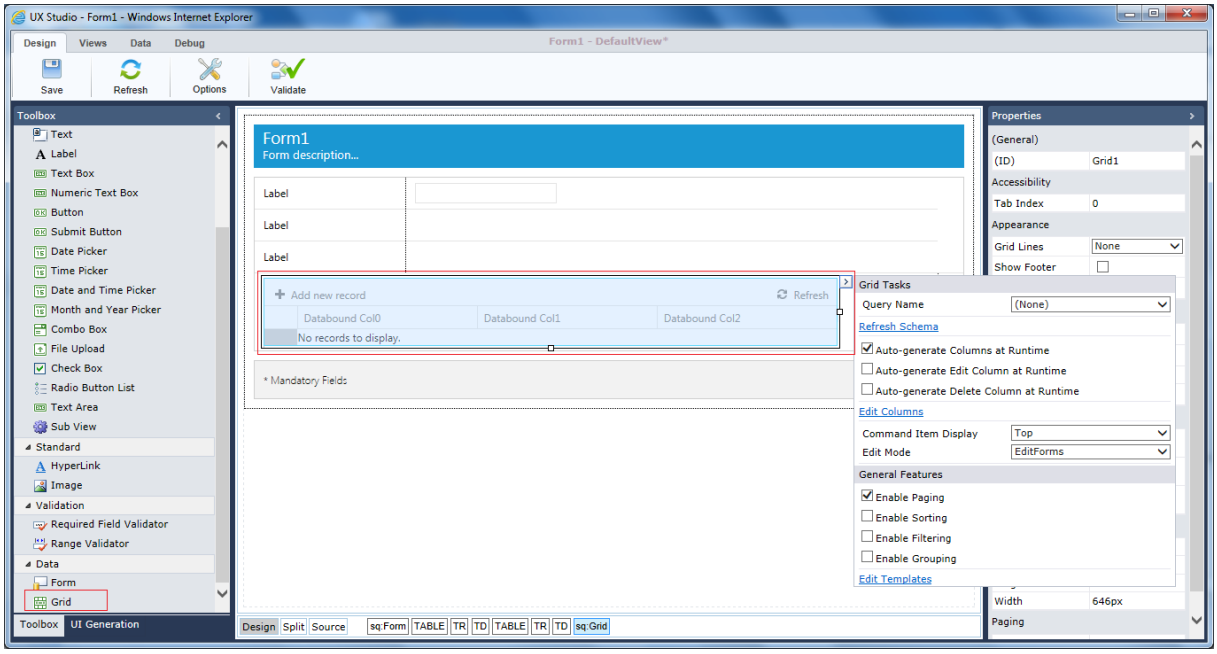
Adding a Grid to a Form

Grids enable you to implement a multi-row form, such as an inventory list. There are several methods of adding a grid to a form. Below, we will describe one method. The video below shows an alternative method:

Adding a Grid to a Form using UI Generation – 5 min. Video

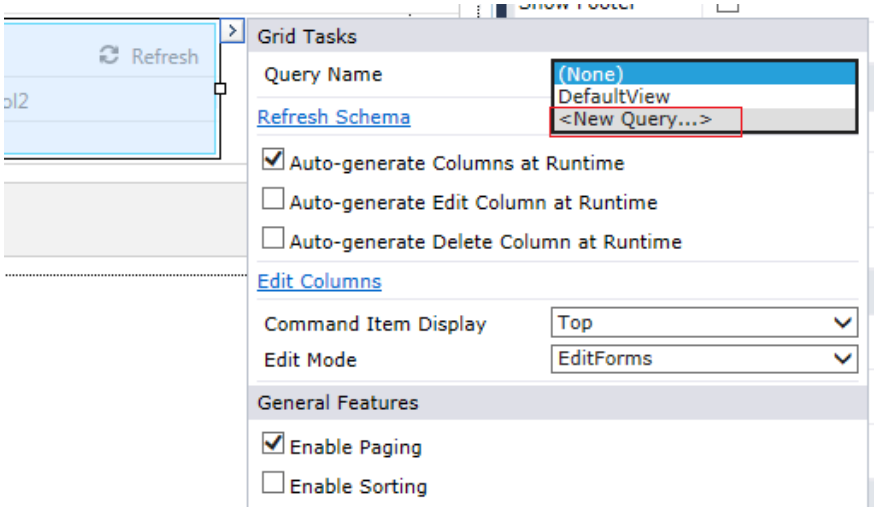
To add a grid to a form:

1. In the UX Studio, from the **Toolbox > Data** section, click the **Grid** control. A grid is added to the canvas.

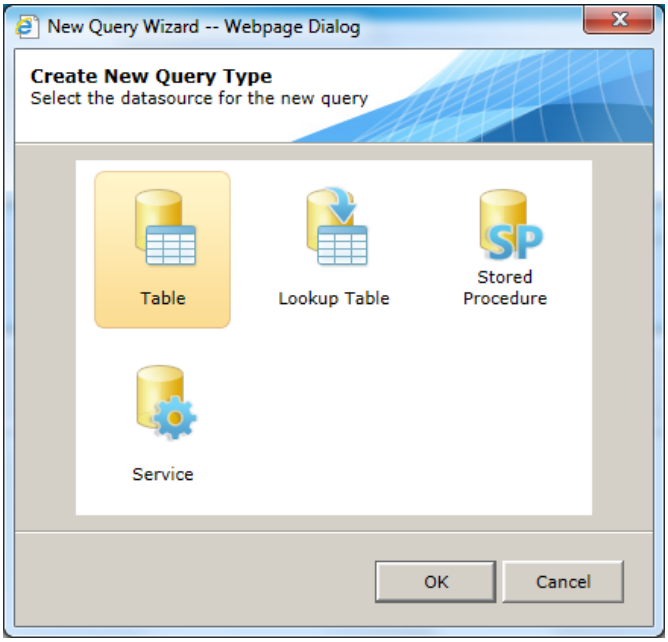


Add Grid to Canvas

2. Open the Grid's easy menu, select **Query Name > New Query** and create a query (e.g. a Table) for the grid data.



New Query



Create New Query

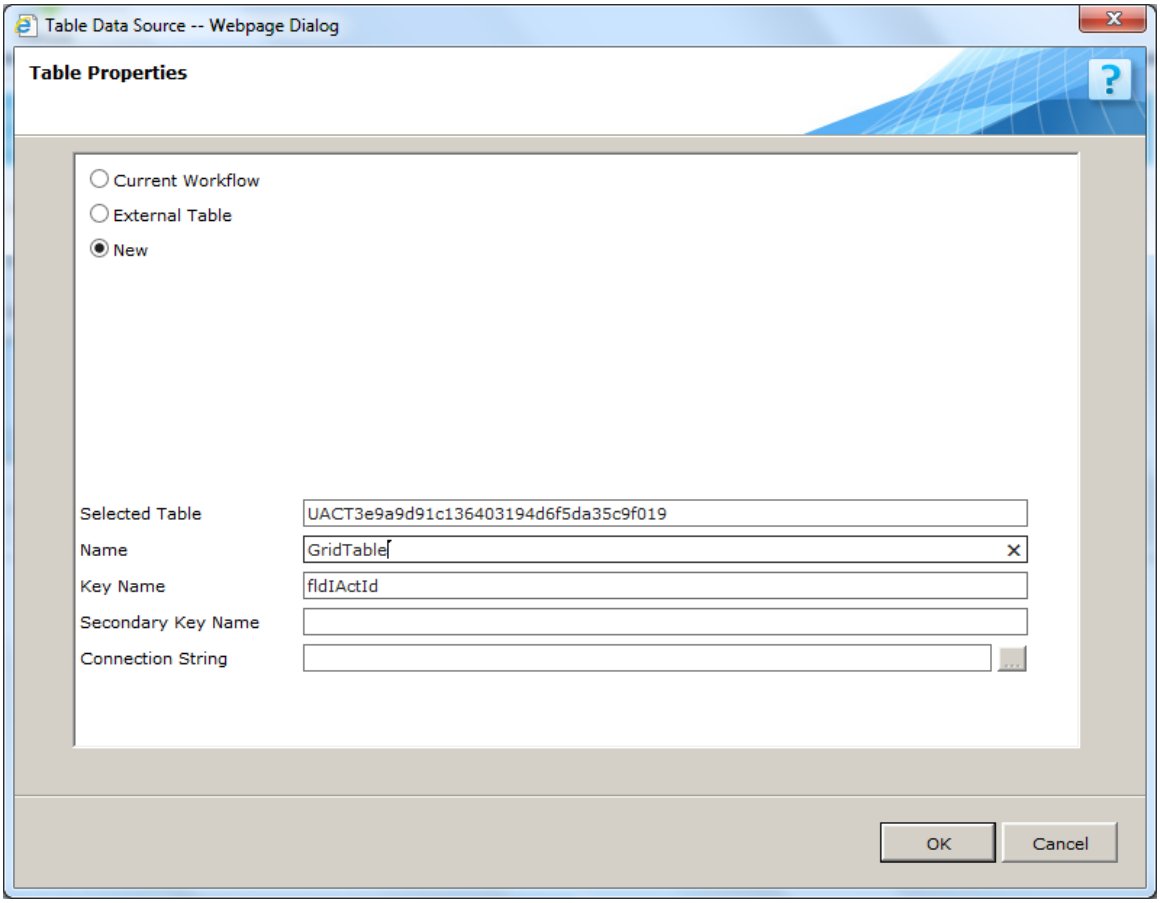
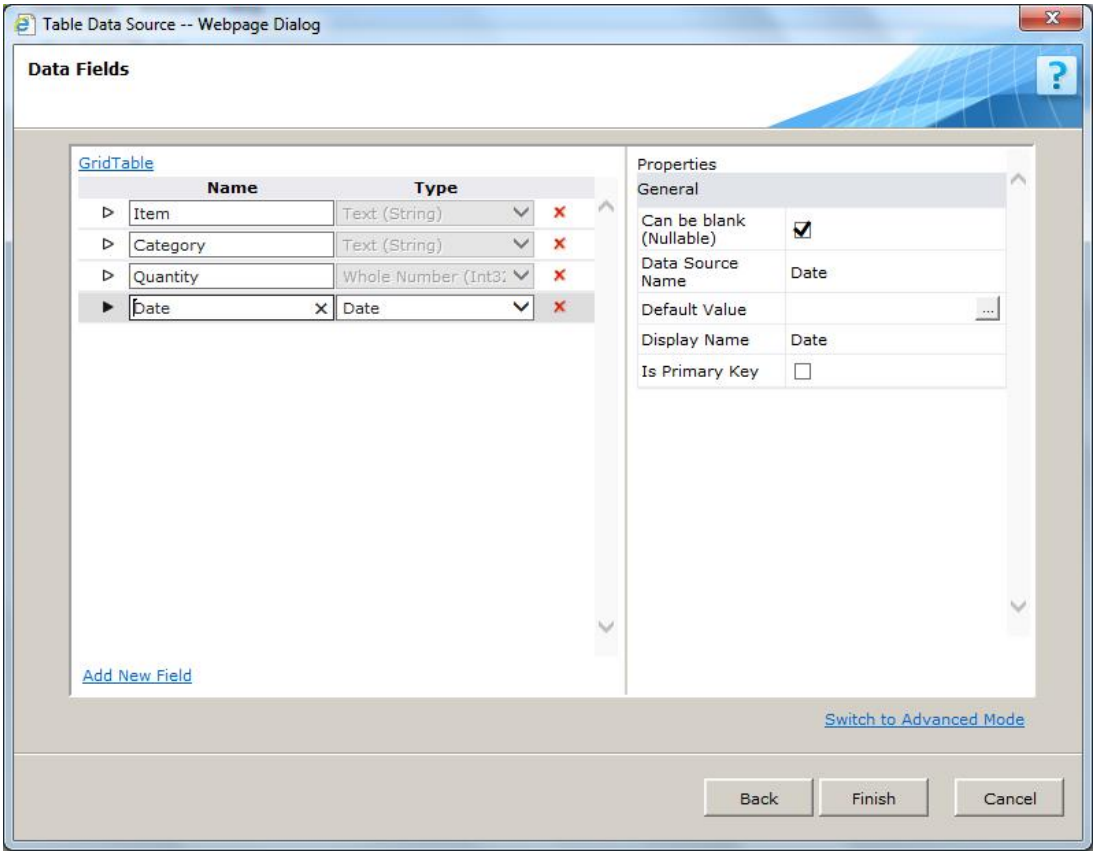


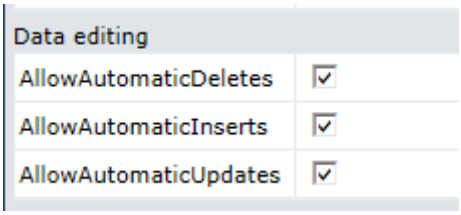
Table Properties

3. From the top ribbon, open the Data Manager (select **Data > Data Model**), select to edit the grid's table and add columns (Data Fields) for the grid:



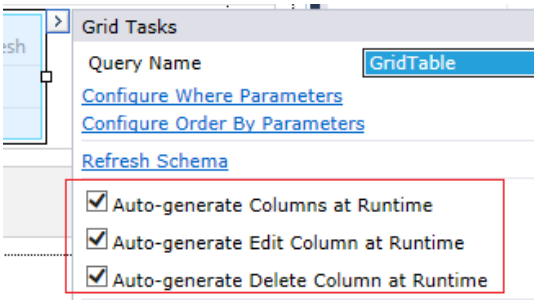
Data Fields

4. From the Grid **Properties > Data editing** section, select: AllowAutomaticDeletes, AllowAutomaticInserts, AllowAutomaticUpdates



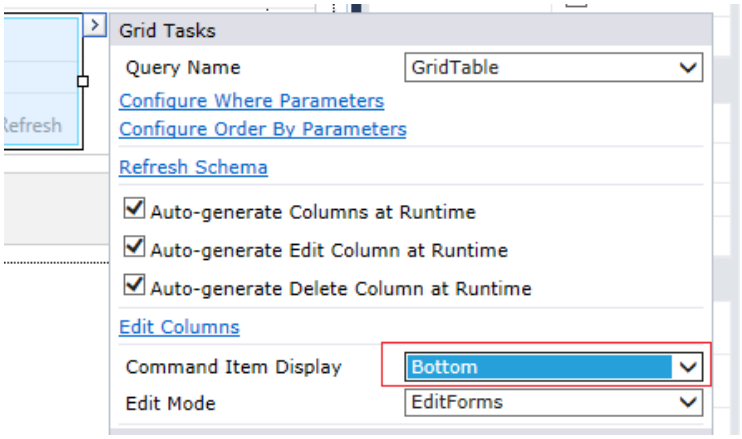
Grid Properties > Data Editing Section

5. From Grid's Easy Menu, select:
   - AutoGenerateDeleteColumn, to give the user a delete grid row option.
   - AutoGenerateEditColumn, to give the user an edit grid row option.



Grid Easy Menu

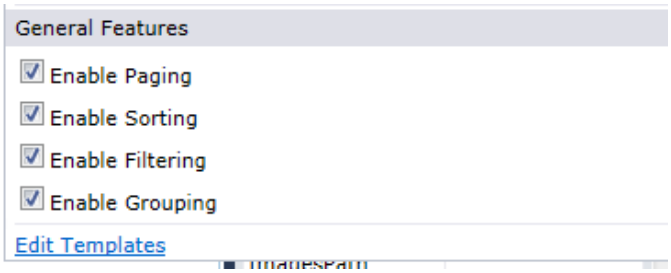
6. (Optional) Modify the location of the **Add new record** button of the grid from the Easy Menu, by selecting options from the **Command Item Display** field. You can also modify the **Edit Mode** for grid rows (In Place, Edit Forms, Popup) here:



Command Item Display

7. (Optional) From the easy menu, select:
   - Enable Paging.
   - Enable Sorting
   - Enable Filtering
   - Enable Grouping

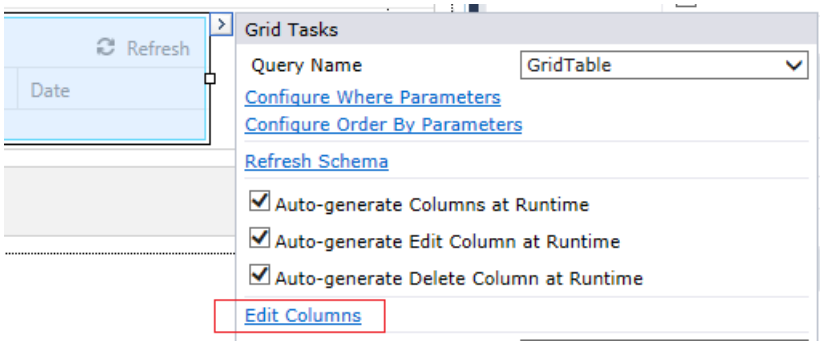
   to enable end users to perform paging, sorting, filtering and grouping of the grid data.



User Grid Data Manipulation Settings

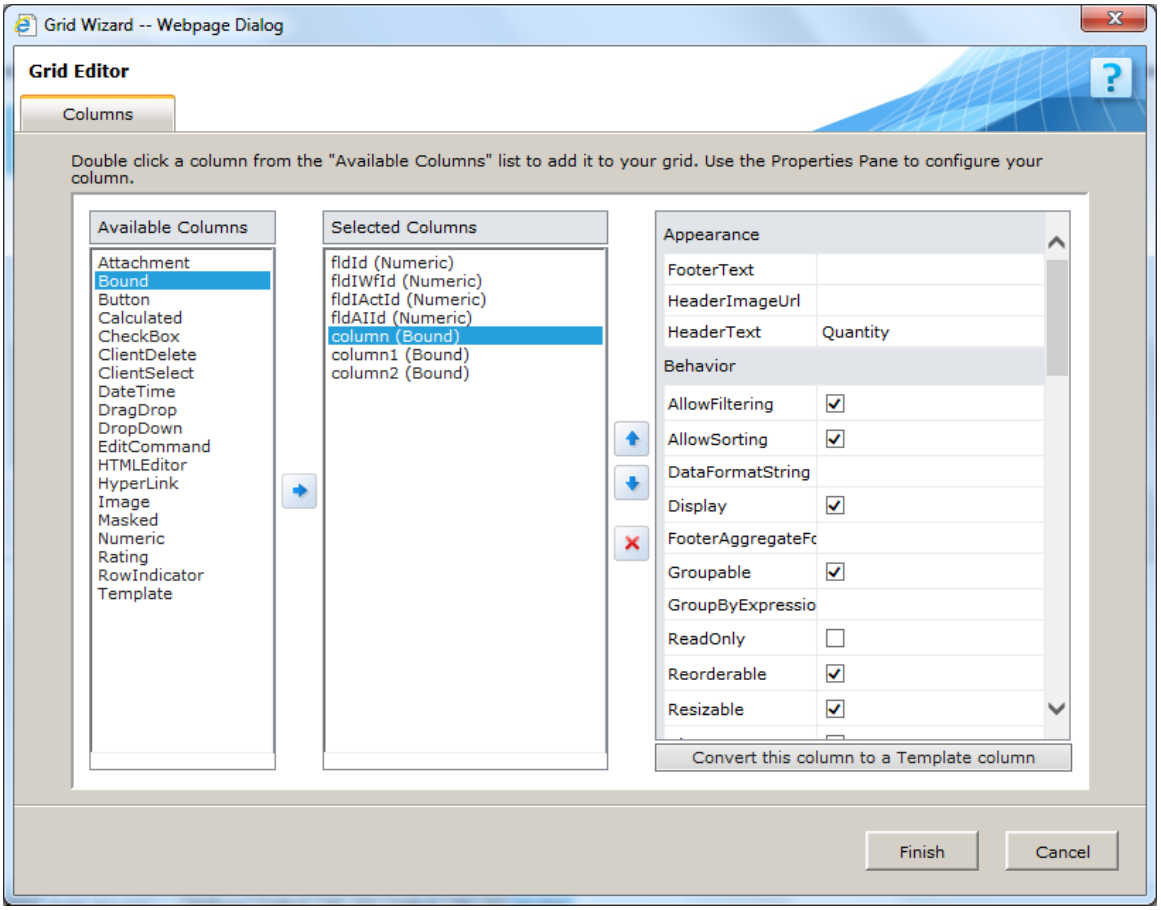
To edit the grid display columns, (for example to add the columns you added as Data Fields and to remove the columns you do not want to display, e.g. fldID, fldiwfID, etc.):

1. Open the Grid easy menu and select **Edit Columns**.



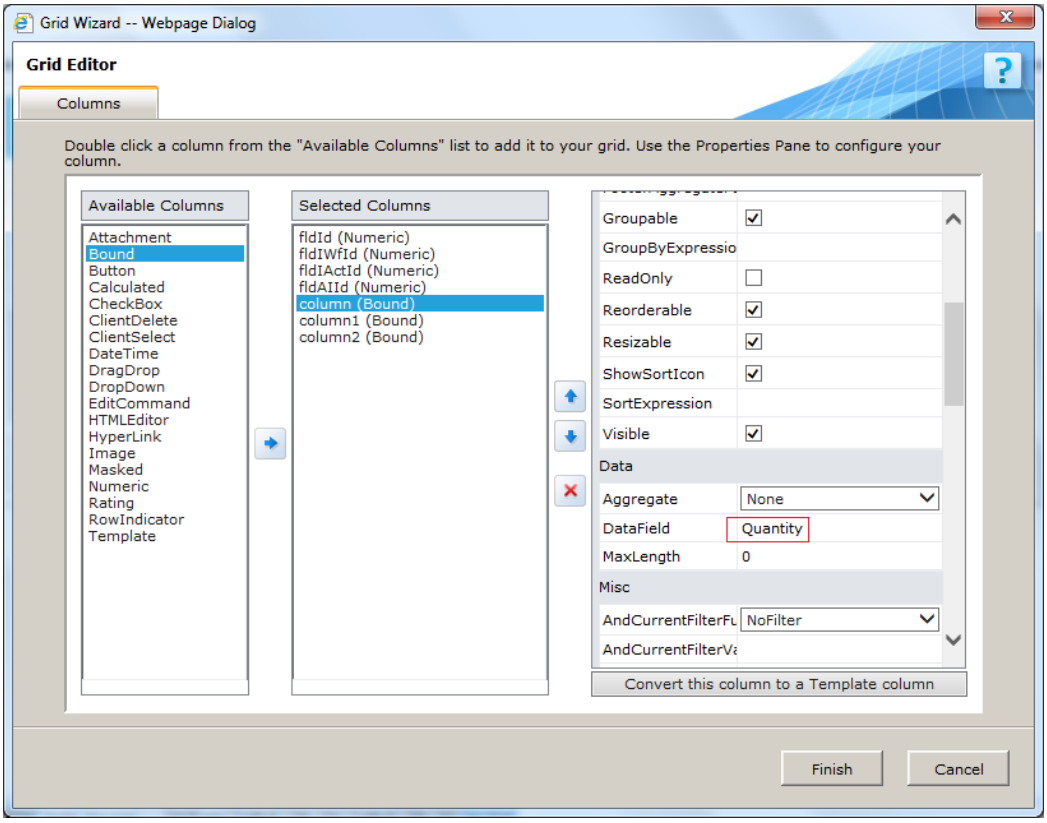
Edit Grid Columns

The *Grid Editor* appears.



Grid Editor

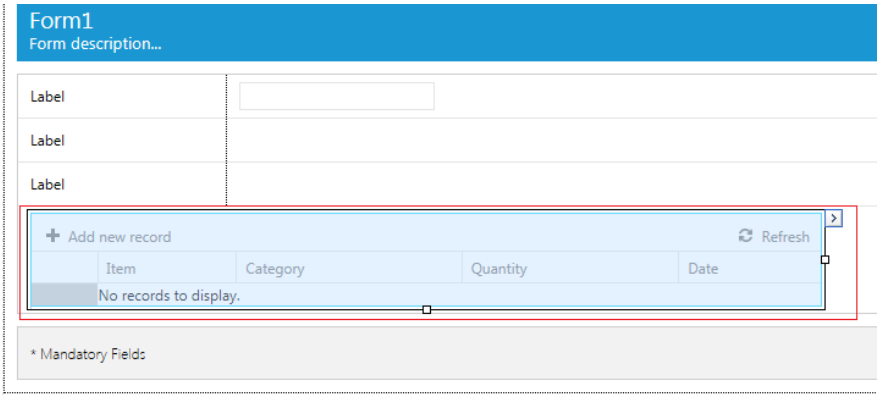
2. To add a column from the Available Columns list, select the column name and click  OR double-click the column name. It moves to the Selected Columns list.
3. To delete a column from the Selected Columns list, click the column name and click  .
4. To move the position of the columns, click the column name and use the  buttons.
5. You can edit column properties in the Properties area on the right.
6. Add several Bound columns and connect them to the data fields you added. You must enter the name of the column in the **Data Field** property of each column. Also change the **Display** property to the name of the column header you wish to display, and the **DataType** property to the correct type.



Adding Data Field Columns

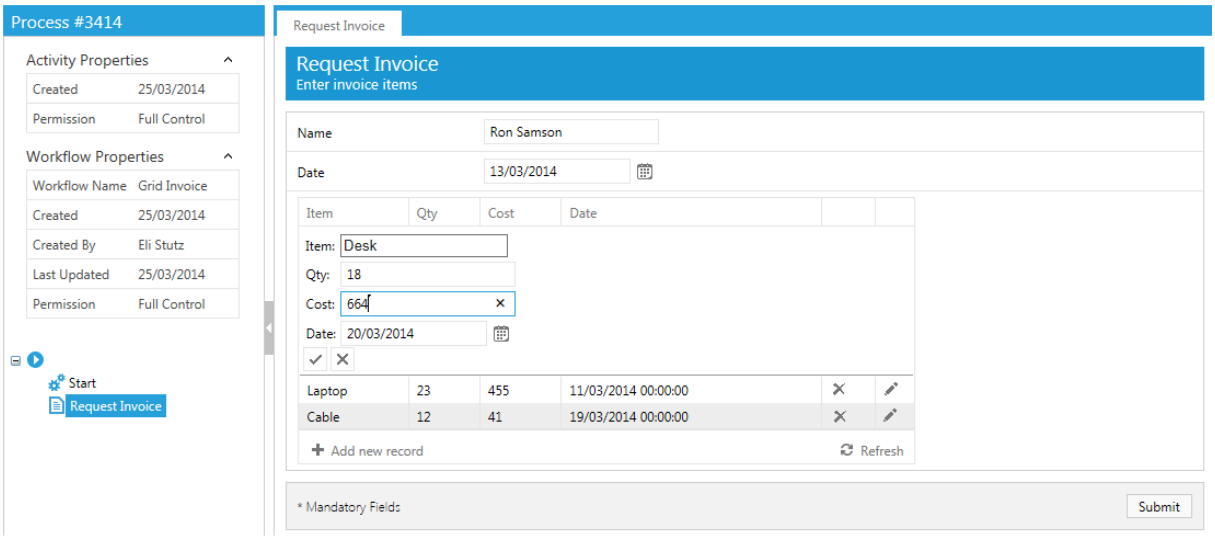
7. (Optional) Click **Convert this column to a Template column**, to reuse the column's settings.

Now your grid should include the columns you have selected.



Grid Columns

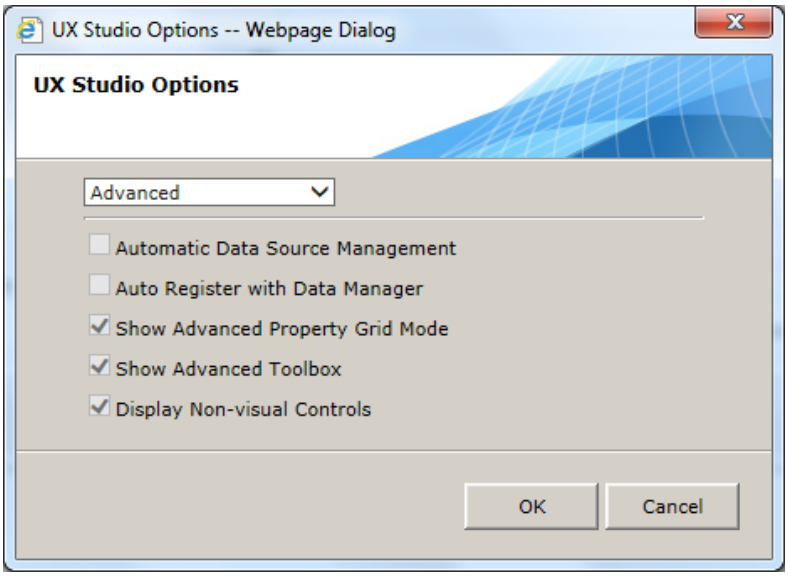
Now you are ready to run or debug your grid form:



Grid in Flowtime

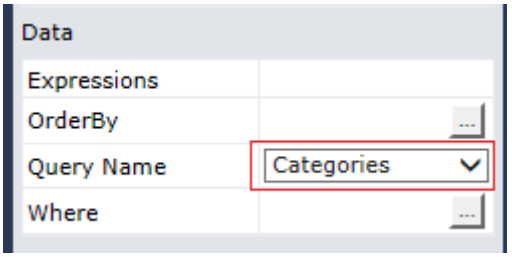
To add a dropdown list as one of your grid columns:

1. In the Data Model (**Data > Data Model**), create or select a Lookup Table query for the data of the dropdown list.
2. Change the view to advanced user view (from the top ribbon, select **Design > Options.** Select **Advanced**).



Advanced View

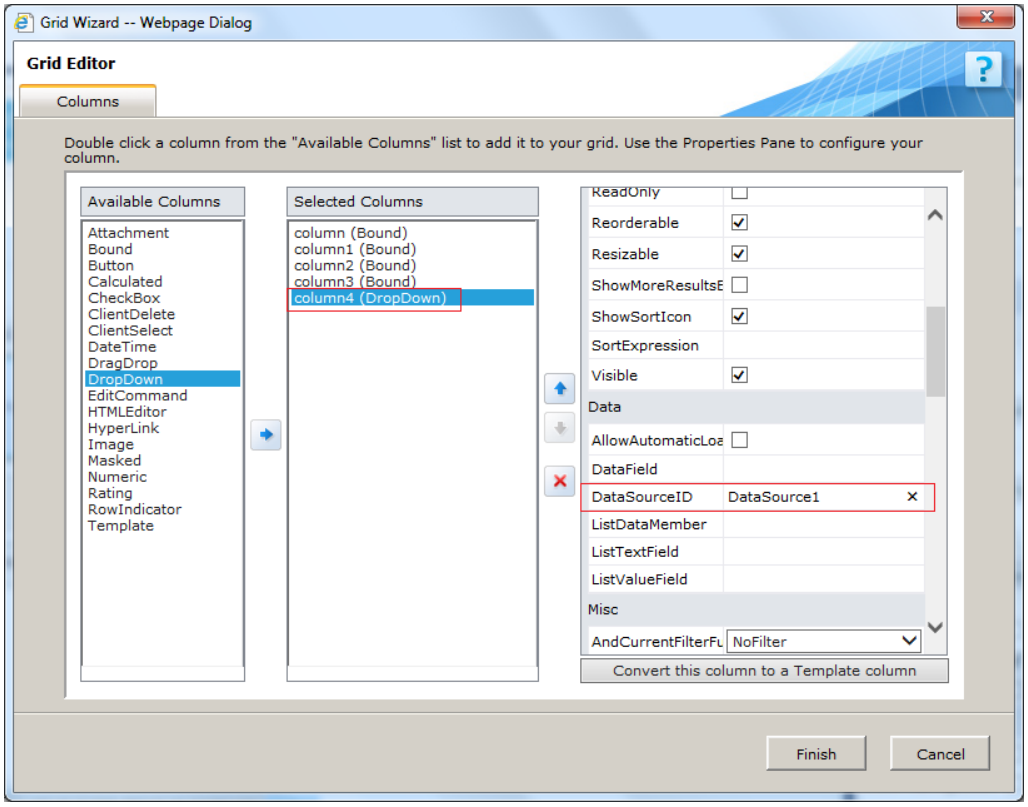
3. From the **Toolbox > Data** section click **Data Source**. A Data Source is added to the form.
4. In the **Entity Data Source > Properties** pane, in the **QueryName** property, select the Lookup Table query (e.g. Categories).



Query Name Property

5. Select **Edit Columns** from the Grid's Easy Menu. Select the dropdown column, and in the DataSourceID attribute, enter the value of the data source you added in step 3.

   For Example : DataSourceID="EntityDataSource2"



DataSourceID

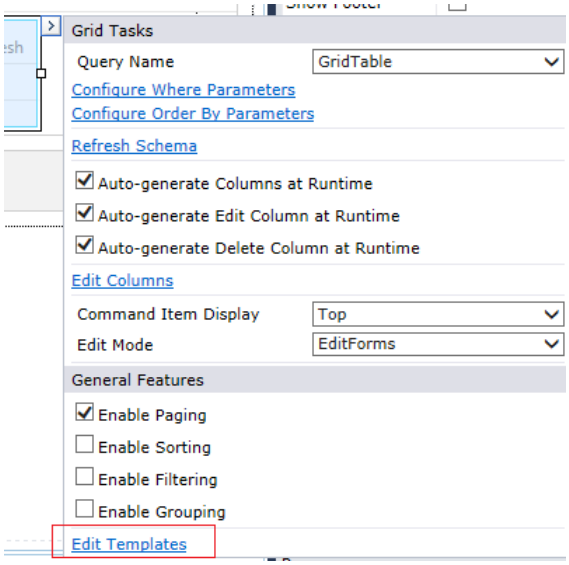
Creating a Custom Grid Add/Edit Record Form

By default, the grid automatically includes an **Add New Record** button, which opens a form where the user can enter a new record by entering the values of the grid display columns. The edit record form also displays these columns.

Alternative, you can create a custom form for a user to add new/edit records of the grid.

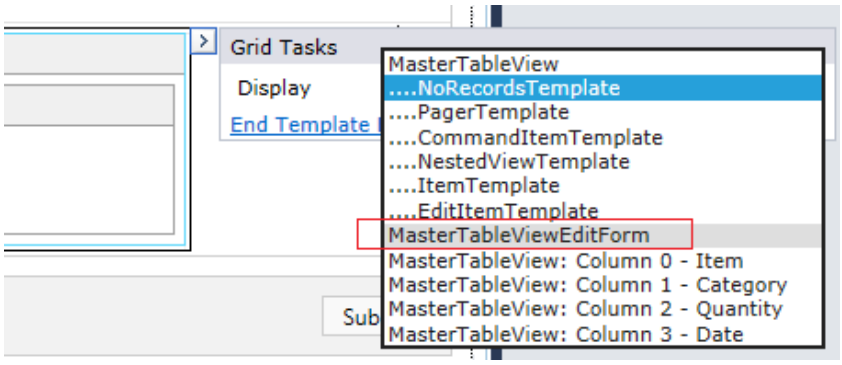
To create a custom add/edit record form that contains different fields than the display columns:

1. From the Grid's easy menu, click **Edit Templates**.



Edit Templates

2. From the dropdown list, select **MastertableViewEditForm**.



Edit Form

3. Edit your custom form, adding values for the fields you want the user to complete when adding a new record/editing a record of the grid. Ensure that each input control is bound to your field in the data source.

Note: For ease of development, you can create a form in your view and copy the markup to MasterTableViewEditForm.

Defining Filtered Grids

The UX Studio enables you to create a filtered grid (filtered with fields on the same form), using a Where Parameter wizard. Click here for how to use this feature.

Grid Features

You can enable several additional grid features by editing the grid markup.

The example markup below presents a grid with several added features (see the color code in the markup below):

Features:

- Enable Ajax on the grid
- Allow sort on the grid columns
- Set the add record position: Top, Bottom, TopAndBottom
- Set the Edit mode, it can be PopUp, InForm, InPlace
- Edit and delete columns with Images
- Images instead of text in the edit\insert form
- Client side reorder animation
- Center the edit form popup to the center of the screen
- Drop Down column

Markup:

```
<script type="text/javascript">
    var popUp;
    function PopUpShowing(sender, eventArgs) {
      popUp = eventArgs.get_popUp();
      var gridWidth = sender.get_element().offsetWidth;
      var gridHeight = sender.get_element().offsetHeight;
      var popUpWidth = popUp.style.width.substr(0,
popUp.style.width.indexOf("px"));
      var popUpHeight = popUp.style.height.substr(0,
popUp.style.height.indexOf("px"));
      popUp.style.left = ((gridWidth - popUpWidth) / 2 +
sender.get_element().offsetLeft).toString() + "px";
      popUp.style.top
=  (sender.get_element().offsetTop/2).toString() + "px";
    }
  </script>

<asp:UpdatePanel ID="UpdatePanel1" runat="server"><ContentTemplate>
<sq:Grid runat="server" ID="Grid2" DataSourceID="Address_DS"
GridLines="None" Width="95%" CellSpacing="0" Culture="English (United
```

```
Kingdom)"  AllowAutomaticDeletes="True" AllowAutomaticInserts="True"
AllowAutomaticUpdates="True" AllowTrue OrderBy="Address"">

<MasterTableView CommandItemDisplay="Bottom" DataKeyNames="fldId"
DataSourceID="Address_DS" EditMode="PopUp"
AutoGenerateColumns="False">

<CommandItemSettings ExportToPdfText="Export to
PDF"></CommandItemSettings>


<RowIndicatorColumn Visible="True" FilterControlAltText="Filter
RowIndicator column"></RowIndicatorColumn>


<ExpandCollapseColumn Visible="True" FilterControlAltText="Filter
ExpandColumn column"></ExpandCollapseColumn>

<Columns>

<sq:GridDropDownColumn DataSourceID="Foreign_DS"
ListTextField="Foreign" ListValueField="fldId" DataField="Foreign"
AllowAutomaticLoadOnDemand="True" HeaderText="Foreign"
SortExpression="Foreign" UniqueName="Foreign"
FilterControlAltText="Filter Foreign column"></sq:GridDropDownColumn>

<sq: GridDropDownColumn DataSourceID="Countries_DS "
ListTextField="Country" ListValueField="fldId" DataField="Country"
HeaderText="Country" SortExpression="Country" UniqueName="Country"
FilterControlAltText="Filter Country column"></sq:GridBoundColumn>

<sq:GridBoundColumn DataField="Address" HeaderText="Address"
SortExpression="Address" UniqueName="Address"
FilterControlAltText="Filter Address column"></sq:GridBoundColumn>

<sq:GridBoundColumn DataField="ComplementAddress"
HeaderText="Complement Address" SortExpression="ComplementAddress"
UniqueName="ComplementAddress" FilterControlAltText="Filter
ComplementAddress column"></sq:GridBoundColumn>

<sq:GridBoundColumn DataField="PostalCode" HeaderText="Postal Code"
SortExpression="Postal Code" UniqueName="PostalCode"
FilterControlAltText="Filter PostalCode column"></sq:GridBoundColumn>

<sq:GridBoundColumn DataField="Locality" HeaderText="Locality"
SortExpression="Locality" UniqueName="Locality"
FilterControlAltText="Filter Locality column"></sq:GridBoundColumn>

<sqr:GridEditCommandColumn ButtonType="ImageButton"
EditImageUrl="~/Shared Resources/images/Orna/Edit.gif"
FilterControlAltText="Filter EditCommandColumn
column"></sqr:GridEditCommandColumn>

<sqr:GridButtonColumn ButtonType="ImageButton" CommandName="Delete"
Text="Delete" ImageUrl="~/Shared Resources/images/Orna/Delete.gif"
UniqueName="DeleteColumn" FilterControlAltText="Filter DeleteColumn
column"></sqr:GridButtonColumn>

</Columns>


<EditFormSettings>
```

```
<EditColumn ButtonType="ImageButton" InsertImageUrl="~/Shared
Resources/images/Orna/AddRecord.gif" UpdateImageUrl="~/Shared
Resources/images/Orna/Update.gif" CancelImageUrl="~/Shared
Resources/images/Orna/Cancel.gif" UniqueName="EditCommandColumn1"
FilterControlAltText="Filter EditCommandColumn column"></EditColumn>
</EditFormSettings>
</MasterTableView>

<FilterMenu EnableImageSprites="False"></FilterMenu>

<ClientSettings AllowDragToGroup="True" AllowColumnsReorder="True"
ReorderColumnsOnClient="True" ColumnsReorderMethod="Reorder">
<Selecting AllowRowSelect="True"></Selecting>

<ClientEvents OnPopUpShowing="PopUpShowing"></ClientEvents>

<Animation AllowColumnReorderAnimation="True"
AllowColumnRevertAnimation="True"></Animation>
</ClientSettings>

<HeaderContextMenu CssClass="GridContextMenu
GridContextMenu_Default"></HeaderContextMenu>
</sq:Grid><sq:BoundControl runat="server" TargetControlID="Grid2"
AutoConfig="True"></sq:BoundControl>

</ContentTemplate>
</asp:UpdatePanel>
<sq:EntityDataSource runat="server" ID=" Address_DS "
QueryName="Addresses" EnableInsert="True"
EnableUpdate="True"></sq:EntityDataSource>
<sq:EntityDataSource runat="server" ID="Countries_DS"
QueryName="Countries" ></sq:EntityDataSource>
```

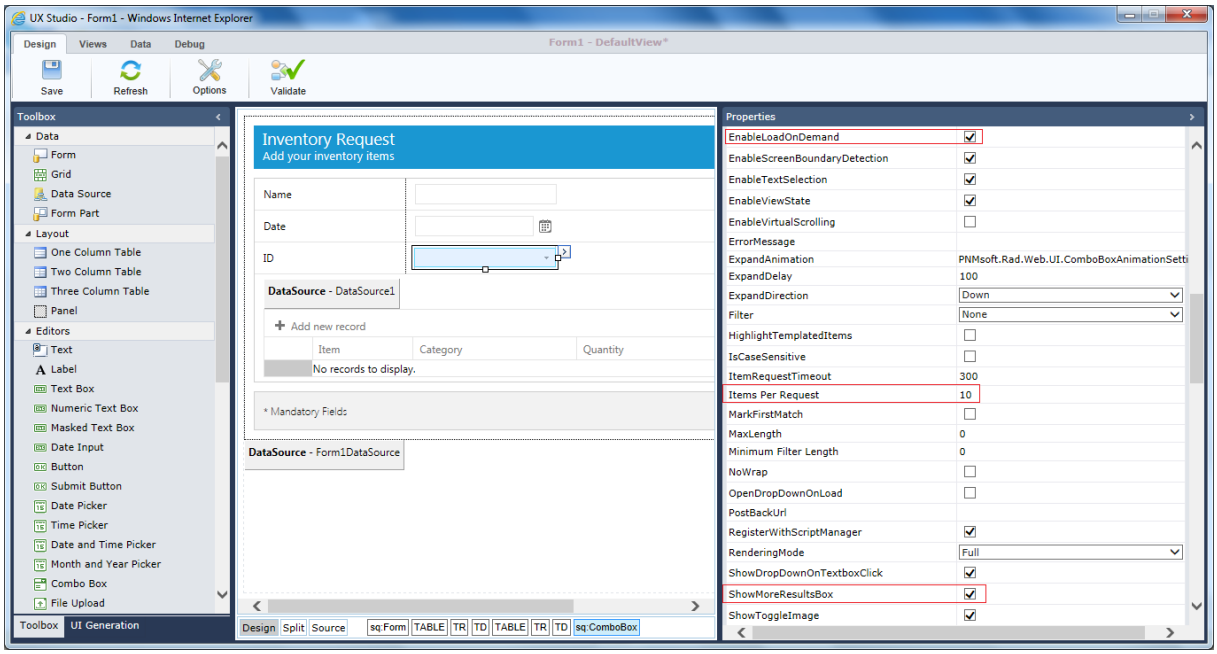
Defining an Advanced Combo Box

The UX Studio provides advanced control over a combo box's appearance and behavior with a set of properties you can define.

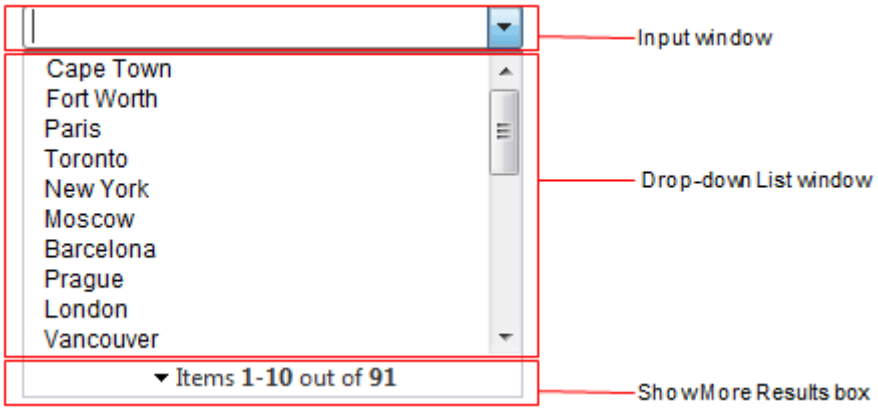
To define an advanced combo box:

1. From the top ribbon > **Options**, select **Advanced** mode.
2. Click the combo box, and from the Properties tab, select the following options:
   - **EnableAutomaticLoadingOnDemand:** enable automatic loading of list results.
   - **ShowMoreResultsBox:** display the Show More Results box.

- **ItemsPerRequest:** define the number of items to display in the drop-down list (e.g. 10 in the figure below).



Combo Box Properties



Advanced Combo Box

You can define additional combo box properties from the Properties pane.

Defining Filtered (Interconnected) Combo Boxes

The UX Studio enables you to create filtered (interconnected) combos. For example, a combo of countries and a combo of cities. When you select the country, the combo of cities is repopulated based on the country selected.

To define filtered (interconnected) combo boxes:

1. Set the first combo as AutoPostBack.
2. Create a function to clear the second combo (e.g. ClearCities in example below).

3. Call this function on OnClientSelectedIndexChanging and OnClientKeyPressing events of the first combo.
4. Set the second combo as EnableAutomaticLoadOnDemand.
5. Add where logic to filter the data source control of the second combo based on the first combo selection (see where expression in example below):
   - For table-based combo boxes, add a "where" expression in the DataSource and a WhereParameters section.
   - For service-based combo boxes, add only a WhereParameters section. Define the service ds parameter value as :@[parameter name in WhereParameter] and the Mask = "In".

For Example:

```
<script type="text/javascript">
function ClearCities(sender, eventArgs)
{
var clientId = $sq("[id$='_CitiesComboBox']")[0].id;
var citiesCombo = $find(clientId);
citiesCombo.get_items().clear();
citiesCombo.set_text("");
}
</script>

<asp:UpdatePanel runat="server"><ContentTemplate>

<sq:ComboBox runat="server"
ID="CountriesComboBox"
DataTextField="Name"
DataValueField="fldId"
AutoPostBack="True"
DataSourceID="CountriesDataSource"
OnClientSelectedIndexChanging="ClearCities"
OnClientKeyPressing="ClearCities"></sq:ComboBox>


<sq:ComboBox runat="server"
ID="CitiesComboBox"
DataTextField="Name"
DataValueField="fldId"
EnableAutomaticLoadOnDemand="True"
DataSourceID="CitiesDataSource">
</sq:ComboBox>

<sq:DataSource runat="server"
ID="CountriesDataSource" QueryName="Countries">
</sq:DataSource>
<sq:DataSource runat="server" ID="CitiesDataSource"
QueryName="Cities"
where="Convert.ToInt32(it[&quot;CountryId&quot;])=Convert.ToInt32(@Countr
yId)" >
<WhereParameters>
<asp:ControlParameter ControlID="CountriesComboBox" Name="CountryId" />
</WhereParameters>
</sq:DataSource>
</ContentTemplate>
</asp:UpdatePanel>
```

Combo Box Caching Items

See Combo Box Load On Demand.

Form Table Editing

A table editing easy menu makes form Table editing easy. To add a row or column to your form, simply select the required action from the easy menu:

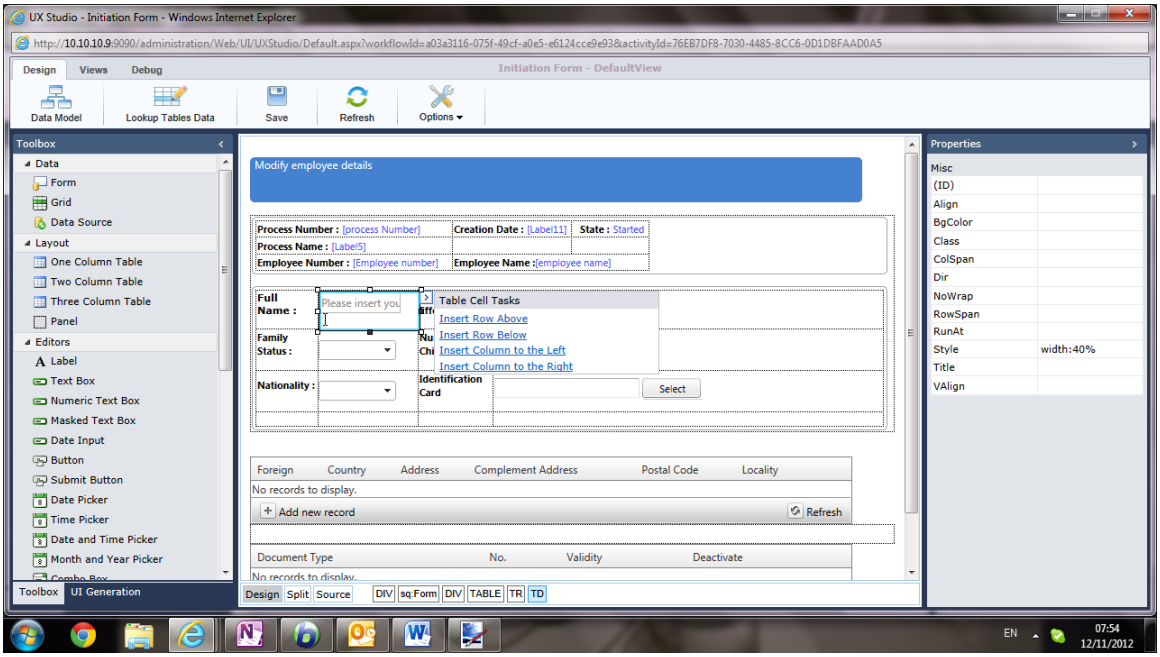


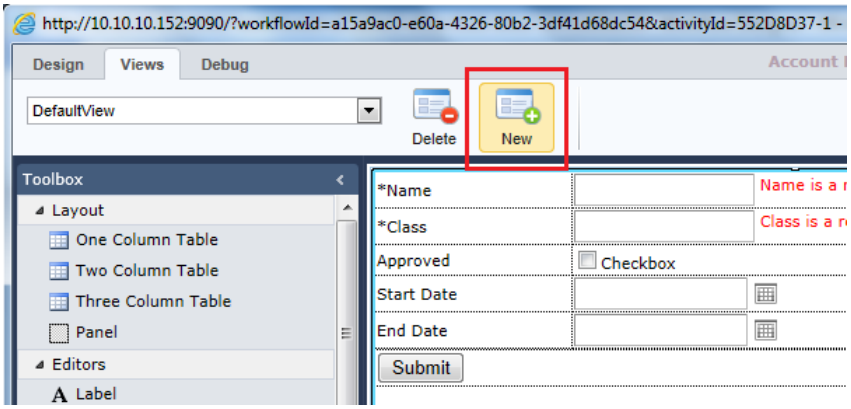
Table Cell Editing

Managing Multiple Views

The UX Studio enables you to create multiple views within the same Template for various platforms (e.g. PC, Mobile). Each of these views access the same data model.

To create a new view:

1. In the UX Studio top ribbon > Views tab, click **New**.



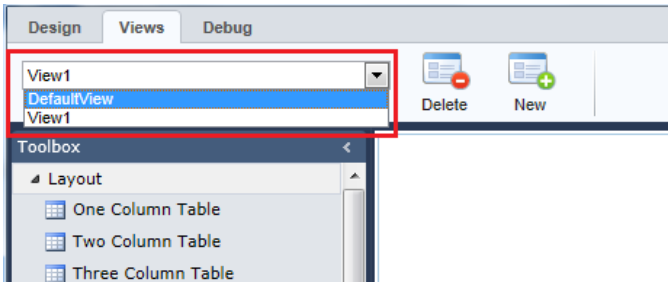
New View

A new view is opened in the UX Studio Canvas, named View1 (the original view is called DefaultView).

Now you can design a separate view according to your needs.

To toggle between views:

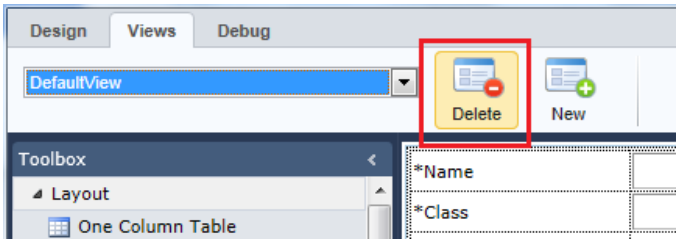
1. In the UX Studio top ribbon > Views tab, select the view of your choice from the dropdown list.



Toggle Between Views

To delete a view:

1. In the UX Studio top ribbon > Views tab, click **Delete**. The selected view is deleted.



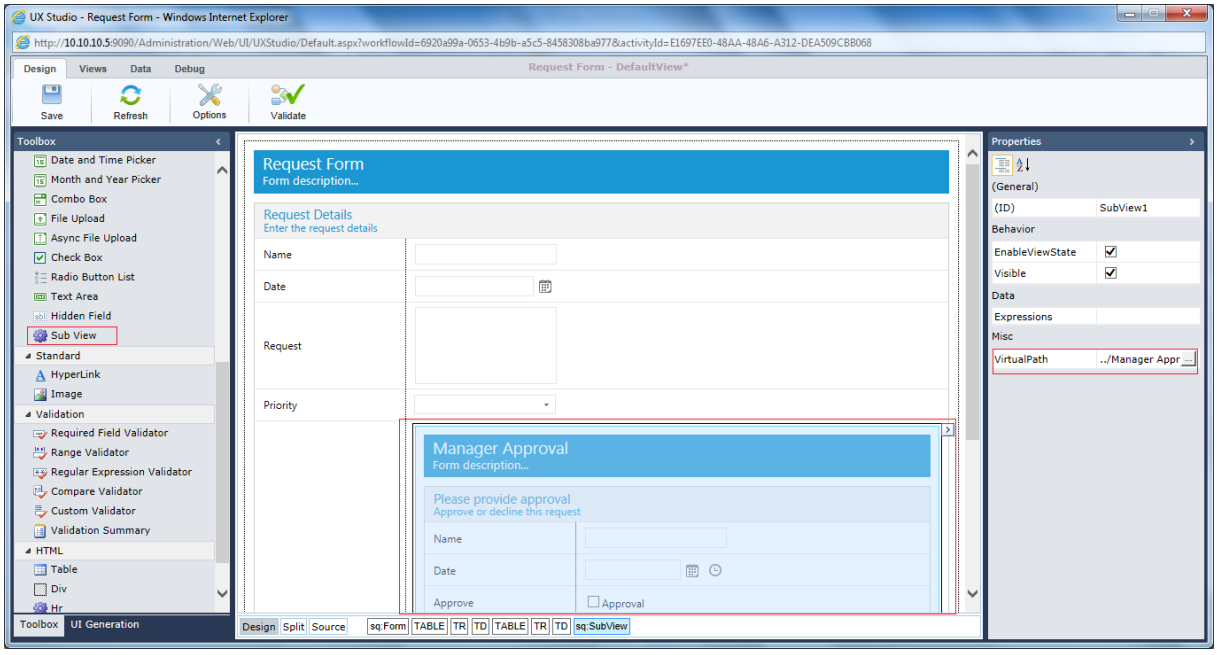
Delete a View

To add a sub view to the current view

1. Create the two views (the main view and the view you wish to add as a sub view – see To create a new view, above).
2. In the current view, from the Toolbox Editors area, click **Sub View**. A Sub View control is added to the UX Studio Canvas.
3. In the Sub View's **Properties** pane, **Misc** section, enter the name of the view you wish to add in the VirtualPath property. Now this view is included as a sub view in the current view.

You can add sub views from the current form, from the other activities in the workflow, or from other workflows. To point Sequence to the correct sub view in other forms, in the sub view properties click **…** to select the sub view from another form, or enter the VirtualPath to the subview as follows:

../../../Space_name/wf_name/Form_name/View_name



Adding a Sub View

Loading Form Sub Views on Demand

When your implementation requires showing and hiding sub-views in the parent view, it is recommended to use one sub view with changed virtual path than many sub-view controls.

Using many sub view controls may result in a very large control tree on the server side which will burden the performance.

The following client side markup and server side code achieves this switch upon button click:

Client-side Markup:

```
<%@ Control Inherits="SequenceEx.Forms.Samples.CustomFormControl, …." %>

<sq:Button runat="server" Text="Switch View" ID="Button1"
OnCommand="OnSwitchButtonCommand" CommandArgument="View2"></sq:Button>

<sq:Button runat="server" Text="Switch View" ID="Button2"
OnCommand="OnSwitchButtonCommand2"></sq:Button>

<sq:SubView runat="server" ID="SubView1"
VirtualPath="View1"></sq:SubView>
```

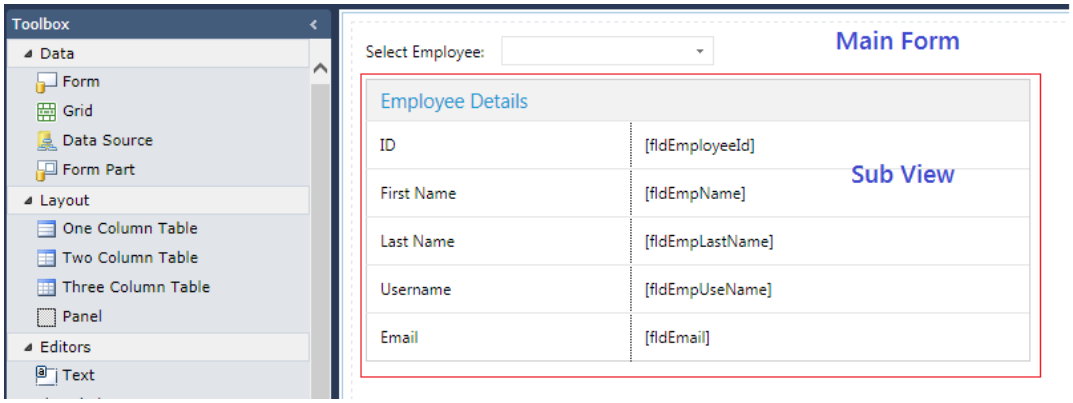
Service-side Code:

```
using System.Web.UI.WebControls;

using PNMsoft.Sequence.Forms.Web.UI;

using Sq = PNMsoft.Sequence.Forms.Web.UI.Controls;

namespace SequenceEx.Forms.Samples

{

public class CustomFormControl : FormControl

{

    protected void OnSwitchButtonCommand(object sender, CommandEventArgs e)

    {

           Sq.SubView subview =
           (Sq.SubView)this.TemplateControl.FindControl("SubView1");

           subview.VirtualPath = (string)e.CommandArgument;

    }

    protected void OnSwitchButtonCommand2(object sender, CommandEventArgs e)

    {

           Sq.SubView subview =
           (Sq.SubView)this.TemplateControl.FindControl("SubView1");

           subview.VirtualPath = "View2";

    }

}

}
```

Passing Parameters from Main Form to Sub View

Note: this feature is available from v7.7 and above.

You can pass parameters from a form to its sub view. For example, you have a sub view which displays records of employees, and in the parent form there is a combo box, where the user can select an employee. You want the sub view to display the details of the selected employee.



Passing Parameters to Sub View

You can do so using the following code:

Main Form:

```
<%@ Control %>

<table class="sqf-main-table" cellpadding="0" cellspacing="0">

 <tr>

  <td>

   <sq:Label runat="server" Text="Select Employee:" Width="100px" />

   <sq:DataSource runat="server" ID="DataSource1"
QueryName="ViewEmployees"></sq:DataSource>

   <sq:ComboBox runat="server" ID="ComboBox1" DataSourceID="DataSource1"
AutoPostBack="true" DataTextField="Name" DataValueField="Id"
EnableAutomaticLoadOnDemand="True" ShowMoreResultsBox="True"
ItemsPerRequest="10"></sq:ComboBox>

  </td>

 </tr>

 <tr>

  <td>

   <sq:SubView runat="server" ID="SubView1" VirtualPath="../UserDetails/DefaultView.ascx"
Skin="Default">

    <Parameters>

     <asp:ControlParameter Name="EmployeeId" ControlID="ComboBox1" DbType="Int32" />

    </Parameters>
```

```
    </sq:SubView>
   </td>
  </tr>
</table>
```

Sub View:

```
<%@ Control %>
<sq:Form runat="server" ID="tblEmployeesForm"  DataKeyNames="fldEmployeeId"
          DataSourceID="tblEmployeesDataSource" ReadOnly="true" CellPadding="0"
CellSpacing="0" Height="217px"><ContentTemplate>
    <table class="sqf-section-table" cellpadding="0" cellspacing="0">
        <tr>
        <td class="sqf-section-header" colspan="2">
          <h2><sq:Text runat="server" Text="Employee Details"></sq:Text></h2>
        </td>
      </tr>
      <tr>
        <td>
          <sq:Label runat="server" Text="ID"  />
        </td>
        <td>
          <sq:Label runat="server" ID="fldEmployeeId"></sq:Label>
          <sq:BindableControl runat="server" TargetControlID="fldEmployeeId"
DataField="fldEmployeeId"></sq:BindableControl>
        </td>
      </tr>
      <tr>
        <td>
          <sq:Label runat="server" Text="First Name"  />
        </td>
        <td>
          <sq:Label runat="server" ID="fldEmpName"></sq:Label>
          <sq:BindableControl runat="server" TargetControlID="fldEmpName"
DataField="fldEmpName"></sq:BindableControl>
        </td>
      </tr>
```

```
        <tr>
          <td>
            <sq:Label runat="server" Text="Last Name"  />
          </td>
          <td>
            <sq:Label runat="server" ID="fldEmpLastName"></sq:Label>
            <sq:BindableControl runat="server" TargetControlID="fldEmpLastName"
DataField="fldEmpLastName"></sq:BindableControl>
          </td>
        </tr>
        <tr>
          <td>
            <sq:Label runat="server" Text="Username"  />
          </td>
          <td>
            <sq:Label runat="server" ID="fldEmpUseName"></sq:Label>
            <sq:BindableControl runat="server" TargetControlID="fldEmpUseName"
DataField="fldEmpUseName"></sq:BindableControl>
          </td>
        </tr>
        <tr>
          <td>
            <sq:Label runat="server" Text="Email"  />
          </td>
          <td>
            <sq:Label runat="server" ID="fldEmail"></sq:Label>
            <sq:BindableControl runat="server" TargetControlID="fldEmail"
DataField="fldEmail"></sq:BindableControl>
          </td>
        </tr>

      </table>

</ContentTemplate>
</sq:Form>
```

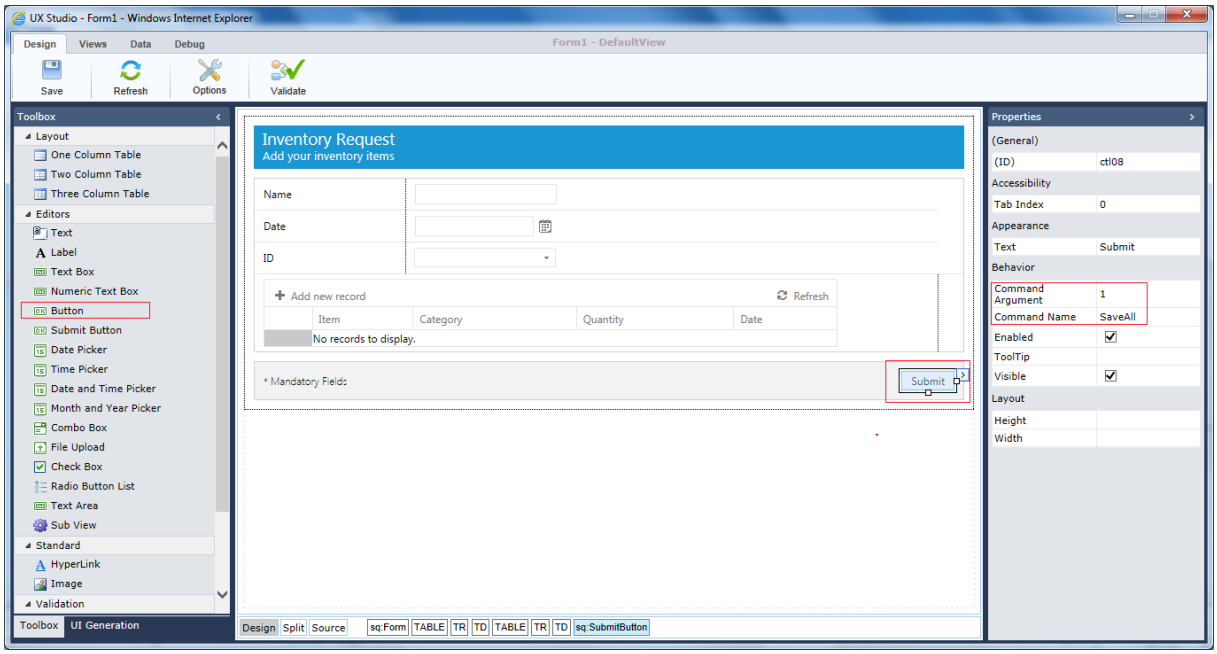
```
<sq:DataSource runat="server" ID="tblEmployeesDataSource"
        QueryName="tblEmployees"  Where="fldEmployeeId == @fldEmployeeId"><WhereParameters>
<sq:TemplateControlParameter PropertyName="Parameters[&amp;quot;EmployeeId&amp;quot;]" Name="fldEmployeeId"></sq:TemplateControlParameter>
</WhereParameters>
</sq:DataSource>
<sq:BoundControl runat="server" TargetControlID="tblEmployeesForm"></sq:BoundControl>
```

Save Command Options

You can add Submit/Update buttons to your forms which enable the user to save the form data. You must give this button a CommandName in the Properties pane.

The UX Studio includes two save CommandName options:

- **Save:** saves all the controls in the current view, but not the controls in its sub views.
- **SaveAll:** saves all controls in this view and in all its sub views.



Save CommandName Options – SaveAll

Note: you can add the Submit Button control which comes with default submit properties already configured (command = SaveAll, text = Submit).

You can also control the behaviour upon clicking the save button using the CommandArgument field. The following CommandArgument options are available:

For Forms:

- 0 – Save but do not Submit.
- 1 – Save and Submit.

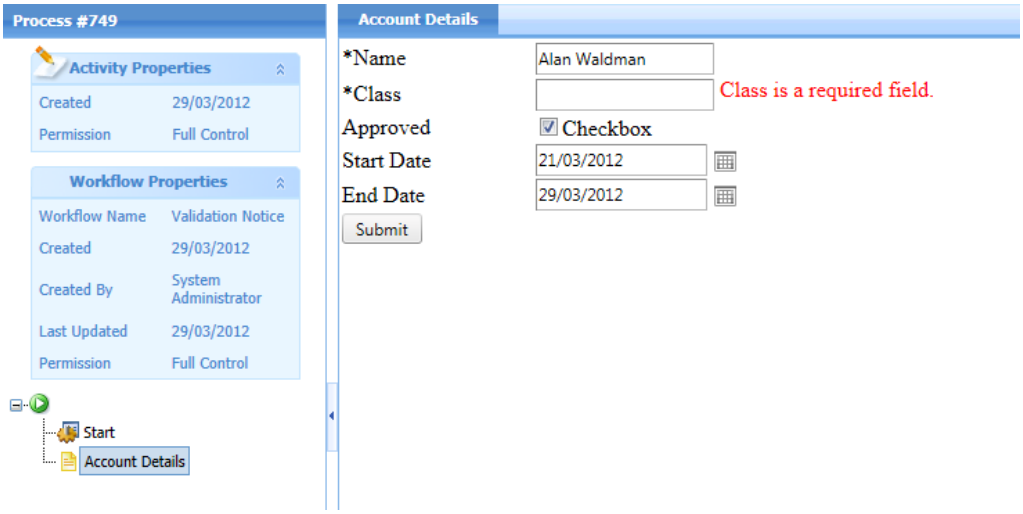
For Tasks:

- 1,0 – Close the task, Save but do not Submit.
- 1,1 – Close the task, Save and Submit.
- 0,0 (this is the same as just writing 0) – Save but do not Close the task or Submit it.
- 1,2 – Set the action items to be closed when the first user submits the task.

Note: 'Submit' here means redirect to the next activity.

Implementing Validations

Sequence Forms use out-of-the-box ASP.NET functionality for validations. ASP.NET contains validation controls and infrastructure which you can use to perform validations for the form. The simplest use for validators is for defining required fields for a user form:



Required Field

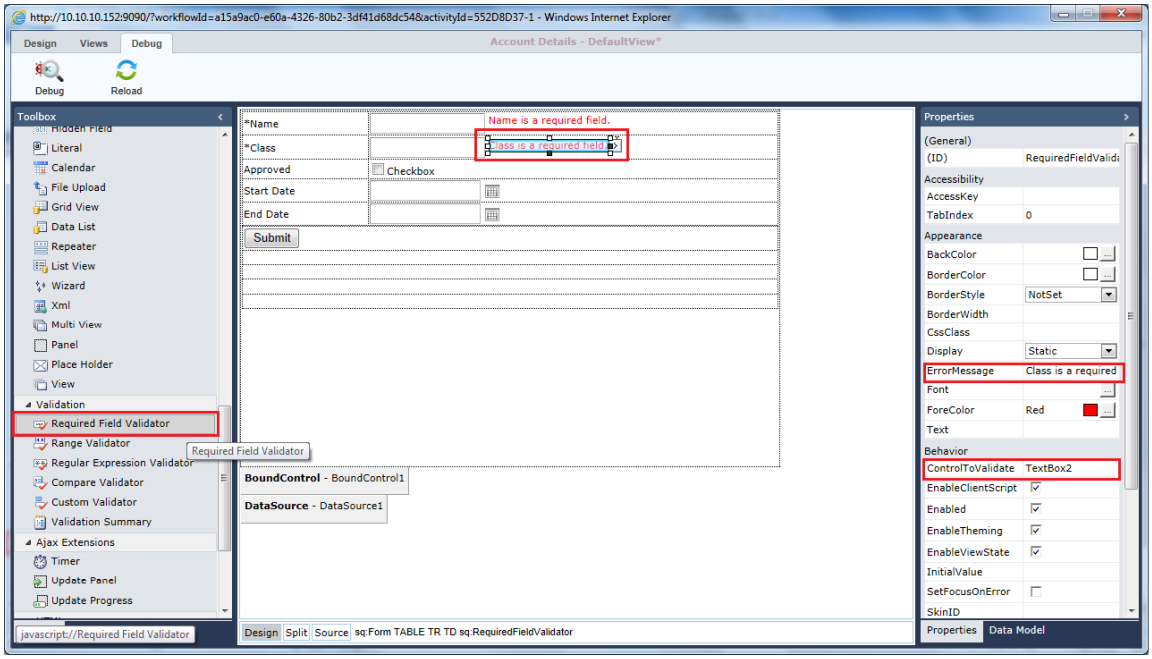
Sequence provides advanced validators as well. The following validation controls are available from the UX Studio Toolbox:

- **Required Field Validator:** provides an error message if the user did not enter a value for this field.
- **Range Validator:** provides a range for validation.
- **Regular Expression:** provides a regular expression validator.
- **Compare Validator:** compares the user value with the provided value.
- **Custom Validator:** enables you to provide custom code for client validation or server side validation.
- **Validation Summary:** use this control to organize all validation messages into one summary.

To add a validator for a control:

1. From the top ribbon > **Options**, select **Advanced** mode.
2. From the **Toolbox**, click the Validator of your choice (e.g. Required Field Validator).

   It is placed on the Form Canvas.
3. In the Properties Pane:
   - Enter the id of the control you want to validate (e.g. Textbox1) in the **ControlToValidate** property.
   - Enter the error message in the **ErrorMessage** property (e.g. 'Name is required. Please enter your name').



Adding Validator Control

4. Define additional properties for the validator (e.g. compare values, operators).

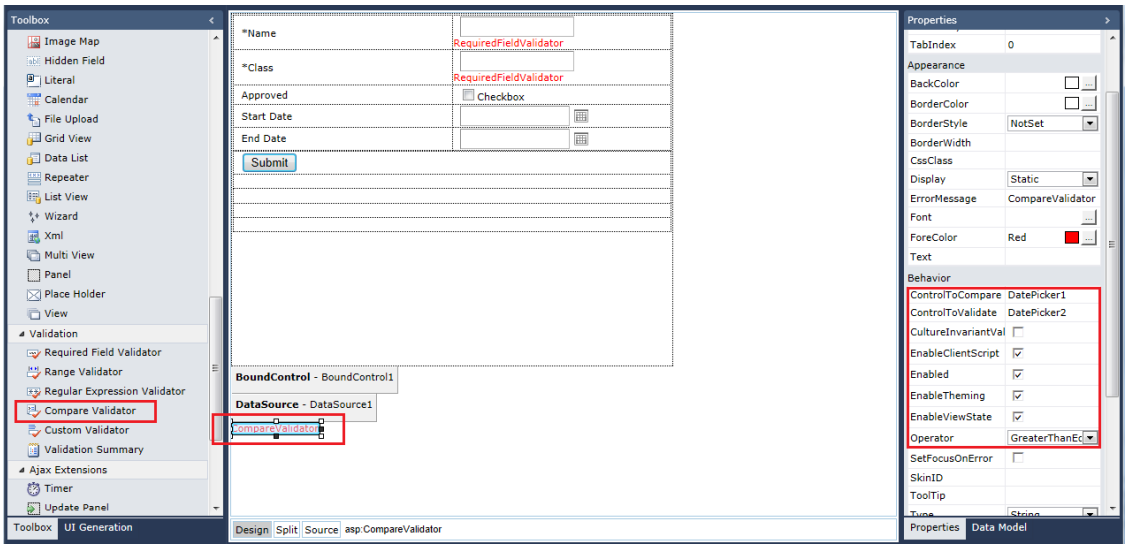
Each validation control can be configured, enabling you to choose how to show the validation, e.g. on the form itself or in a message box. You can also write your own validation controls.

Validation Groups

You can add validation controls to validation groups. Each control in this group will be validated according to the settings of the group.

Example of Compare Validator

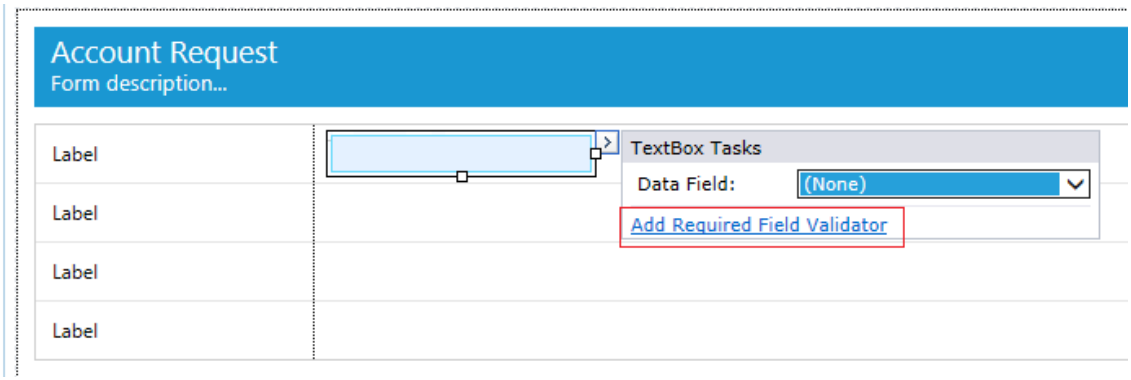
You can add a compare validator to compare two fields on the form, for example, comparing if the end date is greater or equal to the start date:



Compare Validator

Required Field Validator

You can easily set a field as required. To do so, from the easy menu click the **Add Required Field Validator** link. This adds a red asterisk beside the field and enforces it as required before submit. You can change the validation text as required by clicking the asterisk and editing its properties on the right.



Add Required Field Validator

Using Ajax Update Panels

Use AJAX and Update Panels to refresh the page wisely. You may be limited in how often to refresh the page, or which parts of the form to refresh. Therefore, use AJAX technology extensively, as this will improve performance and user experience, and will reduce the weight of the page.

To use AJAX update panels – with Sequence combo boxes, grids or sub views:

1. In the markup, wrap your controls with the following tag:

```
<asp:UpdatePanel runat="server" class="myStyle"><ContentTemplate>
----Controls----
</ContentTemplate> </asp:UpdatePanel>
```