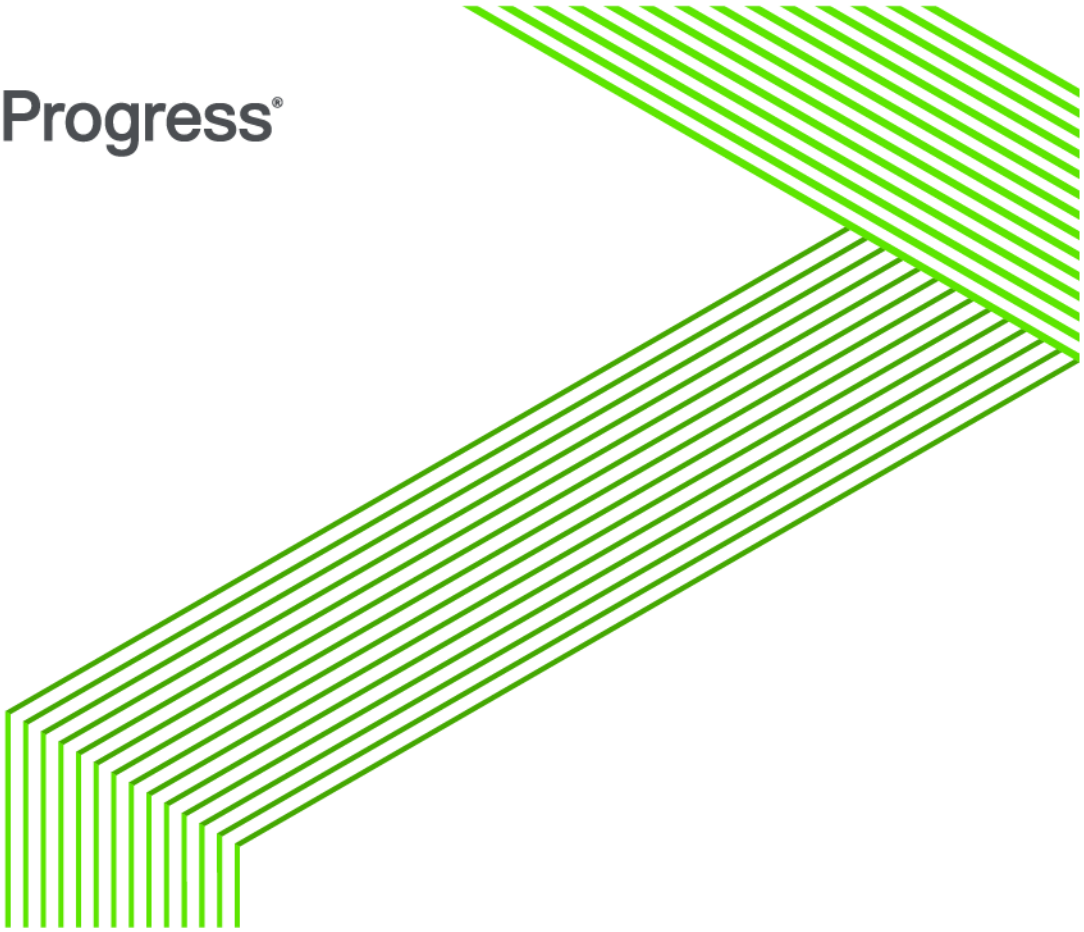
Progress®



Kendo UI® Builder by Progress®:
What's New

Kendo UI® Builder by Progress

Copyright

July 2017

Last updated with new content: Version 2.0

Updated: 2017/07/13

Table of Contents

1

What's new in Kendo UI Builder 2.0

This section describes the new or changed features in Kendo UI Builder 2.0.

For details, see the following topics:

Relaxed OpenEdge installation dependencies

Unlike previous versions, installation of the Kendo UI Builder no longer depends on a current installation of Progress® OpenEdge®. The Kendo UI Builder does continue to support optional run-time integration with Progress Developer Studio for OpenEdge by generating a folder and file structure for published apps that is compatible with a common Web UI project in Developer Studio.

Architecture updates

The Kendo UI Builder architecture has the following changes:

- **Electron foundation** — The architecture is now based on Electron (https://electron.atom.io/) instead of on Node.js alone. Therefore, the Kendo UI Builder Designer runs in a stand-alone Electron window instead of in a Chrome browser page, as in previous versions. This Electron window appears similar to a browser page without an address bar.

  Electron, itself, continues to rely on Node.js, which the Designer uses to preview web apps in your supported default browser with the support of a webpack-dev-server (https://webpack.github.io/docs/webpack-dev-server.html). This webpack-dev-server incrementally updates your previewed app in the browser as you save further changes to it in the Designer using the Kendo UI Generator or in customizable source code in the project using your choice of editor. This minimizes the need to rerun your app preview after making updates to it.

- **Project folder and file structure** — The folder and file structure under the app folder has been changed to support greater extensibility. Some folders and files have been removed or moved under other folders, with some files and folders renamed. The Designer provides an opportunity to migrate existing apps to the new folder and file structure.

Import and migration support

When you first startup the Designer following installation, it automatically imports app projects from your previous installed version of Kendo UI Builder and it prompts with the option to migrate the folder and file structure of these projects to the current version. You can also selectively import Kendo UI Builder app projects from outside of your own project environment, which the Designer also prompts for migration if necessary.

When the Designer migrates an app, it converts the folder and file structure from an earlier version. However, where certain files replace others in the current version, you must manually migrate your custom code from the previous files to the current files where they must reside in the current version.

Login management updates

In previous versions, if a view accessed a data provider that required authentication, the user was prompted for credentials by displaying the login view when they first opened its module.

With this version, if any view accesses a secured data provider, the user is prompted for credentials at app startup, before the landing page is opened. They continue to be prompted for credentials for as many secured data providers as are accessed by views. Only after satisfying every prompt for credentials does the app landing page open.

To support multiple prompts for credentials, the login view is updated to display the defined name of the data provider for which it is requesting credentials.

Preview and build management updates

The app design page and view design page previously supported a toolbar with **Preview** and **Build** options, respectively, to generate and preview an app in the Designer and to generate deployment builds for the app.

In this version, these options have been replaced the following options:

- **Generate** — Generates a build from the latest meta data and custom code suitable for preview in the default browser.
- **Preview** — Previews the latest build in the default browser, after generating an out-dated preview build, if necessary. A currently previewed app is also incrementally updated with source changes without the need to invoke **Generate** or **Preview**. For more information, see Architecture updates on page 8.
- **Publish** — Generates a deployment build from the latest meta data and source code updates that is targeted for a choice of testing (**Debug**) or production (**Release**). Each build target is maintained in a separate subfolder of `build-output` where you can separately update the deployment **Service URI** and **Catalog URI** settings for app data providers. If the application folder (named for your app) containing `build-output` is shared as a Web UI project in Progress Developer Studio for OpenEdge Release 11.7.1 or later, Developer studio can also automatically publish these deployment builds on a schedule that you set.

Data source definition updates

The Add and Edit Data Source dialogs have the following changes:

- **Support for the Lookup semantic type** — A new Lookup semantic type identifies a data source field as a foreign key. A *foreign key* field allows a supported editor type to "look up" a unique record in another (or the same recursive) target (or *parent*) data source based on a single field value that uniquely matches a *candidate* (or *primary*) *key* field in the parent data source. Along with specifying the editor type to perform the look up, you can specify properties that select the parent data source (if more than one supports the same table definition) and a parent data source field for the editor type to display that identifies the matching data. Data-Grid views with forms support foreign key lookups (see Data-Grid-Form and Data-Grid-Separate-Form view updates on page 10).
- **Requirement relaxed to specify a count function** — In previous versions, when you deselect the **Client-side Processing** check box to enable server paging, you must specify a count function that provides the total number of records in the data source result set returned by the server. As of this version, specifying this count function is optional if the data source supports server paging and the Data Object resource defines a count operation for the data source.

Data-Grid view updates

In previous versions, the Data-Grid view was read-only. Now, you can configure the grid in this view to be either read-only or editable, with a choice of three different editing modes:

- **Incell** — Allows editing in the cells of the grid with changes to multiple rows at a time.
- **Inline** — Allows editing in the cells of the grid with changes to one row at a time.
- **Popup** — Allows editing in a popup form with changes to one row at a time.

Each editing mode provides an appropriate set of buttons to add rows for new records, delete records, confirm changes, and cancel changes to the row or rows of the grid.

Note that for Incell editing mode, which supports updates to multiple rows at one time, the Data Object resource that provides the data source must handle changes to all records in a single transaction that succeeds only if all changes are successful.

Data-Grid-Form and Data-Grid-Separate-Form view updates

The Data-Grid-Form and Data-Grid-Separate-Form views have the following updates:

- **Foreign key support** — The new Lookup semantic type identifies a data source field that represents a foreign key defined for the data source. A *foreign key* in the defining data source uniquely identifies a record in a separate target data source table or in the same data source table that defines the foreign key itself (recursive target). The foreign key value in the defining (or *child*) data source record must then match a *candidate* (or *primary*) *key* value in a single record of the target (or *parent*) data source. By including a foreign key field from the child data source as a form field of the view, its editor type "looks up" the identified record in the parent table and returns a field from that record for display in the form without any additional coding required.

  You select the editor type for a foreign key field in the child's Edit Data Source dialog from the options available the Lookup semantic type, which include the combo-box, drop-down-list, and plain-text editor type. At run time, the value of the foreign key field is the value of a single field from the child data source record currently selected in the view. This single field is predefined for the foreign key in the child data source and corresponds to a single field that is also predefined for the candidate key in the parent data source. Along with the editor type, you can also select a parent data source (when more than one have a similarly defined table) and a field in that parent data source to display in the form using the selected editor type.

  At run time, either the combo-box or drop-down-list editor type allows the user to edit the foreign key field value in an editable form by selecting from the list of display field values returned from the parent data source. When the user selects and saves a display field value in the editor type, this changes the underlying foreign key field value in the child data source to the candidate key field value found in the parent record that provides the selected display field. The plain-text editor type always shows the display field value that corresponds to the current foreign key value in read-only mode, even if the form is editable.

  Note that to support a foreign key relationship, both the child and parent data sources must represent top-level tables in their respective Data Object resources. That is, the two data source tables **cannot** participate in a single parent-child data-relation defined in the same Data Object resource.

- **Editor type properties** — When you select an included field in the Form Fields dialog of the view, in addition to the **Label** and **Format** properties that you can specify for the field in previous versions, you can now

specify additional properties that modify the behavior of the selected editor type for that field. The available properties for each editor type are almost identical to the properties available for the corresponding editor components in the user-defined Blank view, except for those properties whose values are already managed by the predefined Data-Grid-* view itself. Note also that most of the properties for any foreign-key field editor type are already set from the existing property settings for that field in its data source definition using the Add/Edit Data Source dialog.

- **Field groups on forms** — In the Form Fields dialog, you can create labeled field groups into which you drag and drop included fields. At run time, these field groups display as labeled tab folders on the form with tabs in the order of the field groups, and with the fields displayed in the order you arrange them in each field group.

New predefined Data-Grid views for related data sources

This version of the Designer provides two new predefined Data-Grid views for displaying data for two data source tables in a parent-child relationship:

- **Stacked-Data-Grids view** — Displays two grids on a page, where the first grid displays records from a parent data source and the second grid displays records from a related child data source directly below the parent grid. The parent grid is read-only. When the user selects a parent row, the related records from the child data source are displayed in the rows of the child grid, which can be either read-only or editable, similar to the Data-Grid view (see Data-Grid view updates on page 10). This view allows the user to display child rows for only one parent row at a time.
- **Hierarchical-Data-Grid view** — Displays a single hierarchical grid structure on a page containing both parent and child rows. Initially, the grid displays only parent rows, which are read-only. When the user clicks an expander on any parent row, a child grid opens indented under the parent row and displays records from the related child data source. The user can individually expand multiple parent rows, which allows the user to display multiple child grids under their respective parent rows at a time. The child grids can all be either read-only or editable, similar to the Data-Grid view (see Data-Grid view updates on page 10). For editable child grids, the user edits and saves row changes for only one child grid at a time.

In the Designer, both views support the same set of properties to separately control behavior of the parent and child grids and rows.

Note that both the parent and child data sources in these views **must** represent tables that participate in a single parent-child data-relation defined in the same Data Object resource.

Blank view updates

Updates to the Blank view include the following:

- **Layout components** — As in previous versions, you can create a layout for a Blank view by first adding rows, then adding columns to the rows, then adding content and navigation components (as well as other rows) to the columns. With this version, you can also first add rows, then directly add content and navigation components to the rows, which automatically creates a column for each component that you add. Adding components directly to rows like this is easier and automatically formats the layout for them. However, you have more control over the layout by adding columns to rows first, before adding all other components to the columns.

- **Data management components** — The editing modes for the Grid component now include **Incell** in addition to the **Inline** and **Popup** modes previously supported, making the editing support for this component similar to that of the Data-Grid view in this release (see Data-Grid view updates on page 10).
- **Editor components** — Include the following new and updated components:
  - **Auto Complete** — A new text box that automatically suggests options for a value based on the user's typed input.
  - **Boolean Radio Button List** — A new component that provides the features of the Radio Button List in previous versions, which is a radio button set with two possible boolean values (true or false).
  - **Radio Button List** — An existing component that is updated to provide a radio button set with multiple values based on the instances of a specified field in the data source result set. This component binds to both a data source instance and a model to provide field values for the buttons in the list. The selected button represents the value of the specified field from the corresponding record.
- **Chart components** — A new category of components that chart data from a single data source, including:
  - **Area Charts** — Charts that plot one or more series of item values as the area under a line plotted along the item points for each series. The line and area for each item series gets a distinguishing color.
  - **Bar Charts** — Charts that plot one or more series of item values as either horizontal bars or vertical columns (depending on the chart type) plotted along the item points for each series. The bar or column for each item series gets a distinguishing color.
  - **Donut Charts** — Charts that plot one or more series of item values as segments of one or more concentric donuts, where each donut plots items for a single series. The segments for each item series therefore occupy one of the concentric donuts. The size of each segment in a given donut occupies space that is sized relative to the item value it represents in that series.
  - **Line Charts** — Charts that plot one or more series of item values as a line plotted along the item points for each series. The line for each item series gets a distinguishing color.
  - **Pie Charts** — Charts that plot item values from one or more series as slices of a single pie. The size of each slice is relative to the item value it represents in its series.
- **Navigation components** — Include the following new and updated components:
  - **Expander** — A new collapsible container that when clicked, alternately expands to display, and collapses to hide its contents. To add contents, you drag at least one layout row to the expander, then drag additional components into that row, or into other rows that you have added. You can add one or more expanders to an existing expander to create a single tree view.
  - **Tab Strip** — A new container for one or more tabs that when clicked, each opens its own container to display contents for access. To add contents, you drag at least one layout row to each tab container, then drag additional components into that row, or into other rows that you have added. You can specify the position as well as the labeling of the tabs in a given tab strip.
  - **Toolbar** — An updated component that provides different types of items, such as buttons, toggle buttons, split buttons, button groups, and other customized elements. The Toolbar Items dialog is updated with a more intuitive UI to more easily identify the different items you can add and the properties you can set for them. In addition, button groups are new in this version. You can create one or more button groups and add buttons to each group. At run time, these button groups appear in the toolbar as drop-down lists of buttons.

Extensibility improvements

Existing Data-Grid* views and the Grid component of the Blank view all support additional event functions that you can code. For example:

- For the form fields of Data-Grid views with editable forms, you can specify a Change event function that executes when a field value changes, among some others, depending on the selected editor type.
- For all grids in all views, you can specify a new Data Bound event function that executes when data is bound to the grid.
- For all editable grids in all views you can specify new Row Create, Row Update, and Row Delete event functions that execute when the respective row changes occur.

The new predefined Data-Grid views for related data sources and some new components of the Blank view also have additional event functions to handle aspects of their behavior.

Miscellaneous UI improvements

The Designer includes a number of additional UI changes and improvements. Following are some of these changes:

- **Updated gear control options on an app card or list item** — In previous versions, this gear control provides the option either to edit the app identification information or to delete the app from the dashboard. With this version, the gear control provides options to duplicate the app (with a different name), remove the app from the dashboard (leaving its folders and files in the current app location), and to permanently delete the app (removing all of its files and folders from the system). You can later import any app that you only remove from the dashboard (see Import and migration support on page 8). Note also that the option to edit the app identification information is now available only through the pencil icon to the right of the app name on the app design page after you click the app card or list item.
- **Selectable color schemes** — A new **Theme** property is available when you first create or edit the app identification information for an app. This allows you to specify or change the color scheme for the app.
- **Run-time labels for modules and views** — In addition to the **Name** property that you cannot change once defined, all module and view definitions now provide a **Label** property where you can specify an alternate label to identify the module or view at run time. If you do not specify a value for this property, the value of the **Name** property is used to label the module or view at run time as in previous versions.
- **Drag-and-drop for modules and views** — You can now change the relative run-time order of modules as they appear on the app landing page and views as they appear in a given module's view list. You can change the order of modules by dragging and dropping them on the app design page. You can change the relative order of views by dragging and dropping views within the **VIEWS** pane on the view design page for a given module.
- **altRowTemplate support in grids** — In addition to properties to specify formatting for the rows of a grid (the **Row Template Function** and **Row Template Id** properties for both the Data-Grid* views and the Blank view Grid component), this version supports properties to format every other row of a grid (the **Alt Row Template Function** and **Alt Row Template Id** properties).