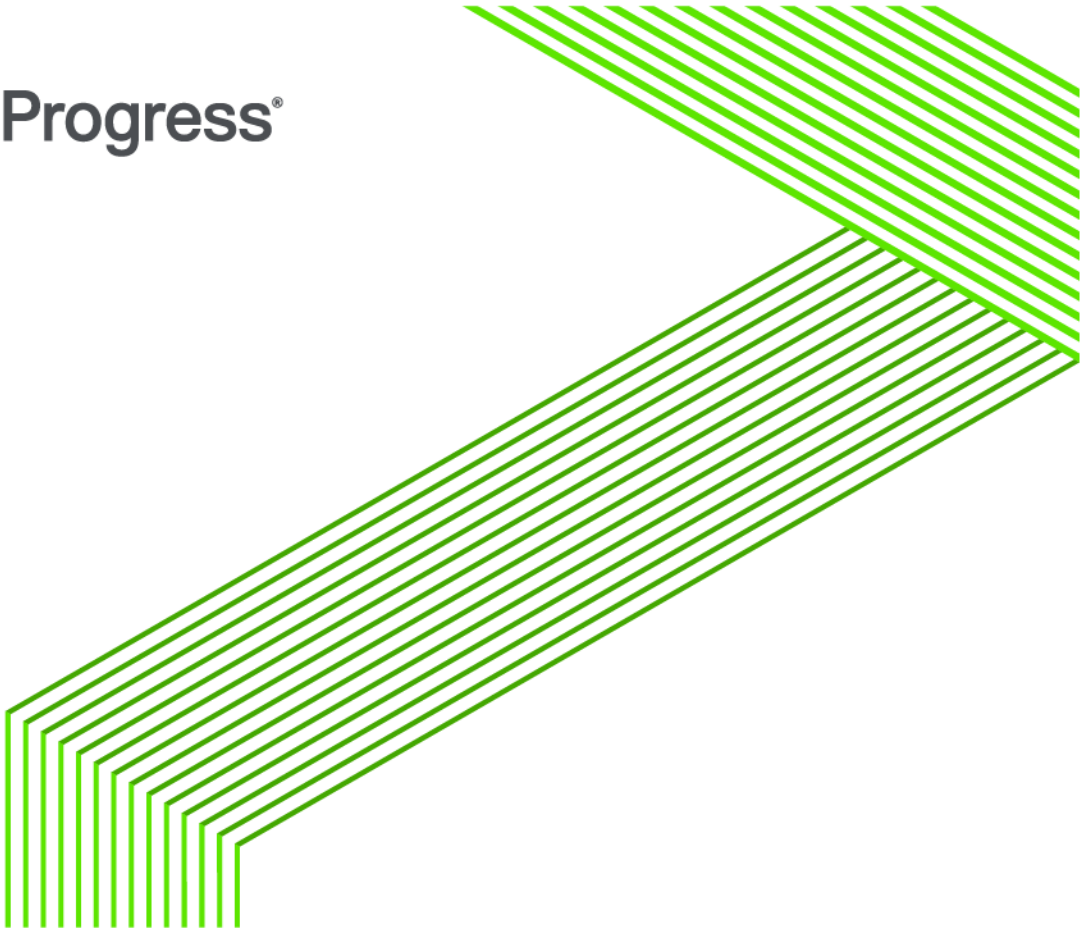
Progress®



Kendo UI® Builder by Progress®:
Using Kendo UI Designer

Kendo UI® Builder by Progress

Notices

November 2016

Last updated with new content: Version 1.1

Table of Contents

1

Introduction to the Kendo UI Designer

The Kendo UI Designer is part of Kendo UI® Builder by Progress®, an application that facilitates modernizing OpenEdge applications by creating web-based UIs. You can use the Designer to design the UIs for your application using a collection of predefined, data-driven views, or you can create custom layouts by dragging and dropping components into a blank view. For more information about the entire Kendo UI Builder package, see Kendo UI Builder by Progress: Modernizing OpenEdge Applications. For a sample workflow for modernizing an application, see Kendo UI Builder by Progress: Sample Workflow.

When you first open the Kendo UI Designer, you're presented with a list of web apps you've previously created, as well as options to:

- display existing apps in a card or list view
- search apps by keyword
- sort apps alphabetically
- edit or delete an existing app (by clicking the gear icon in the card or list item)
- create a new app

You can access this home screen at any time by clicking the Progress logo in the upper-left corner of the screen.

2

Creating an app

To create an app using the Kendo UI Designer, you need to create several sub-components:

- *Data providers*: Data services that provide access to data and associated logic. Each data provider is represented by one service URI.
- *Data sources*: Single tables found within data providers.
- *Modules*: The building blocks of your app, representing a single business function (e.g., order entry or inventory), which are then packaged into a single app. One module is provided for you at creation: it contains a customizable log in screen and a landing page.
- *Views*: The screens that make up each module. There are two types of views: data-driven, composed of predefined templates that the Designer populates with your data sources, and custom (called a Blank view), which allows you to create your own layout.

For more information about these terms, see *Kendo UI Builder by Progress: Modernizing OpenEdge Applications.*

For details, see the following topics:

Creating an app: first step

When you click the **Create App** button in the upper-left corner of the home screen, the Create App dialog box appears. Enter the following information in this dialog:

1. **App name**: The name of your application. The name cannot contain spaces. Only letters, numbers, dashes, and underscores are permitted.
2. **Location**: The location for the files associated with your app. This should match your workspace and project folder of the associated Web UI project in Progress® Developer's Studio for OpenEdge® (PDSOE).
3. **Description**: An optional description of your app. This text will appear next to your app on the home screen.

When finished, click **Create App**, which will take you to the main editing screen.

Note: You can also start editing a pre-existing app by simply clicking on the card or row representing the app, depending on the view. The gear menu on this screen allows you to **Delete**. The **Edit** option does NOT provide access to the full set of app editing tools, but rather allows you to add a brief description and add a logo to the app. You access the same edit dialog you see when clicking on the pencil icon next to the app name in the main editing screen. See Main editing screen on page 10 for more information.

See also

Main editing screen on page 10

Main editing screen

Once you've created a new app (or selected **Edit** from an existing app's drop-down menu), you'll enter the main editing screen. From this screen, you can:

- Click on the pencil icon next to the app name to:
  - Add an optional description of the app for your own reference (not visible to the app user).
  - Add a logo that will appear in the header of the main page of the app. The name of the logo you enter must be the name of an image file located in *`application-folder`*`/src/assets/images`.
- **Preview**: Show a preview of what the finished app looks like and how it behaves. Preview runs the latest build of the web app. You will be given the option to to select **Build & Preview** if you want to build before running the web app.
- **Build**: Generates the HTML and JavaScript source code for the app.
- Add and edit data providers.
- Add and edit modules.
- Edit the Application module, which contains views for a log in screen and a landing page.

See also

Adding data providers on page 11

Adding data providers

To add a data provider:

1. Click **Add Data Provider** on the main editing screen.
2. Enter a **Name** for the data provider. The name cannot contain spaces. Only letters, numbers, dashes, and underscores are permitted.
3. Enter the **Service URI**. This is the URI of the web application that hosts the Data Object Service. For example, your service URI may look something like `http://Your-IP-Address:8980/MyMobileWebApp`.
4. Enter the **Catalog URI**. This is the location of the catalog file generated when the Data Object Service is built on the server. See the help system in Progress Developer's Studio for OpenEdge (PDSOE) for more information.

   Note: Once entered and saved, the name, service URI, and catalog URI of a data provider cannot be further edited.

5. Choose the security model for the service URI from the **Authentication Model** menu. This should match the Authentication of the server for the Data Object Service.
6. (Optional) You have the option to manually create data sources once you've finished creating your data provider, but you can also select **Auto-create Data Sources** to allow the Designer to automatically generate your data sources based on the resources defined in the catalog. A *data source* corresponds to a single table found in a data provider. A data source is created using the schema definition in the catalog, and when auto-create is selected, data sources are automatically generated for each of the following cases, given that the resource is enabled for at least one CRUD operation:
   - A single table: A data source is created using the resource name of the table from the catalog file.
   - A ProDataSet with a single, top-level table: A data source is created using the resource name of the top-level table from the catalog file
   - A ProDataSet with more than one top-level table: A data source for each top-level table is created with the following naming convention: resource name of the ProDataSet + "." + table name.
7. (Optional) Select **Create Data Sources for child tables** to allow the Designer to automatically generate data sources for child tables for a ProDataSet with related tables. These are named resource name + "." + table name. You must select **Auto-create Data Sources** to enable this option.

With one or more data providers created, you can now create data sources using those providers.

See also

Adding data sources on page 11

Adding data sources

To add a data source from an existing data provider:

1. Click **Edit** next to the given data provider in the main editing screen. You will be brought to the Edit Data Provider page
2. Click **Add Data Source**.

Note: If you chose the optional **Auto-create Data Sources** when creating your data provider, some of the work described below will be done for you in this dialog, but you can still manually edit the data source and individual properties and labels as needed. You can also create custom data sources in addition to the ones automatically created.

---

3. Enter a **Name** for the data source.
4. Search for or navigate to the table your app will draw data from. Double-click on the table name, and all of the available fields will appear in the **Included fields** box.
5. You can use the **Include all** and **Exclude all** buttons to move the entire list of fields back and forth between **Included Fields** and **Excluded Fields** lists. You can also drag and drop single fields back and forth between the lists. The order that the fields appear in the **Included Fields** list is the same order that will be used for the columns and/or form fields, so drag and drop the included fields into the order you prefer.
6. When you select any particular field, you can specify the **Label** and **Editor Type**. The label is used as the column head and/or the field label in the view. (See the sections on modules and views for more information.) You can select an editor type based on the underlying semantic type for the given field. For example, a date field might offer the option of a text input editor or a date picker. For more information on semantic types, see *Kendo UI Builder by Progress: Modernizing OpenEdge Applications.*

Note: Enter `plain-text` as the **Editor Type** to display a field as read-only on an otherwise editable form. This is useful for auto-generated fields, such as record identifier fields.

---

7. (Optional) Select **Client-side processing** to indicate your app will retrieve all data before the view is visualized and should perform data presentation operations (sorting, filtering, and paging operations) on the client side. Otherwise, only a page of data is retrieved as the page is displayed, and these operations are performed on the server side.

Note: If client-side processing is not checked, the Business Entity needs to include code for processing the request. This is done by enhancing the business logic to include code filtering and counting the data. Business Entities provide the JSON Filter Pattern (JFP) for the READ method and code for a count method. (You are prompted to enter the name of the count method if **Client-side processing** is unchecked.) See "Updating Business Entities for access by Telerik DataSources and Rollbase external objects" in *OpenEdge Development: Web Services* for an example implementation of the pattern and count method.

Note that if you create one or more views using your data source and then come back to change some of the details of that data source, e.g., labels, editor types, those changes will NOT be reflected in the views you've already created. The changes will be in effect for any new views you create, but those changes will not be propagated back to views already created.

The data source(s) you created are now available for use in a module.

See also

Adding and editing modules on page 13

Adding and editing modules

Once you have an accessible data source, you can create one or more modules to interact with that information:

1. From the main editing screen, click **Add Module**.
2. Add a **Name** for the module. The name cannot contain spaces. Only letters, numbers, dashes, and underscores are permitted.

   Note: Once entered and saved, the name of the module cannot be edited.

3. Expand the **Icon/Color** menu and select an icon, icon color, and background color.
4. (Optional) Add a brief description. This will appear under the module's name on the main editing page and as a tooltip for the module in the generated web app.

You can change some of the information (except for the module name) later on by clicking the pencil icon next to the module in the main editing screen.

Once you've created the module, it will be added to the list of modules on the main editing page. Click **Edit** to enter the module editing screen. Once there, you can:

- add and edit views
- save a view in progress
- revert a view to its saved state
- preview your application
- build your application (i.e., generate the HTML and JavaScript source code)

The next step in creating your module is to add one or more views.

See also

Main editing screen on page 10
Adding and editing views on page 13

Adding and editing views

The Designer provides several types of views to use in your application:

- the built-in Application module contains two views, a login page and a landing page
- the modules you create can contain one or more views that you can customize

The login and landing page views in the Application module are added for you. To add views to your own modules:

1. From the module editing screen, click **Add** in the upper left corner.
2. Add a **Name** for the view. The name cannot contain spaces. Only letters, numbers, dashes, and underscores are permitted.

Note: The name you enter here is what will appear in the navigation panel in the running app. The name you enter while actually editing the view (e.g., **Grid Name**, **Data Form name**, etc.) is for your internal use only.

3. Choose the type of view. What follows is a basic summary, although each type has different properties and behaviors that can be customized, as described in later topics:

   - **Data-Grid**: A read-only, table-like display of the information in the data source. The data source itself determines the organization and display type of the data
   - **Data-Grid-Form**: A split screen featuring a Data-Grid as described above and a form display showing the information in a selected row. This view can be read-only or user editable. If editing is enabled, users edit information in the form display.
   - **Data-Grid-Separate-Form**: Similar to the Data-Grid-Form, but in this case, only the Data-Grid is shown unless users select a row. The information from that row is then shown as a form display on a separate screen. If editing is enabled, the screen switches to a form display suitable for editing.
   - **Blank view**: Design your own custom view by dragging-and-dropping rows and columns to create a layout and then adding functional components to the layout.

   Note: For all views, you are provided with a drop-down menu at the top of the screen that provides an approximate preview (without building the app) of what any given view will look like on a desktop, laptop, tablet in landscape, or tablet in portrait. This can be particularly helpful in finding out how rows will break when viewed on different devices.

4. (Optional) Event functions: After initially creating a view, you can then edit the view by clicking the gear icon next to the view's name and selecting **Edit**. This gives you the ability to indicate three custom functions to handle events:

   - **Init Event Function**: fires the first time the view renders.
   - **Show Event Function**: fires after the view is rendered.
   - **Hide Event Function**: fires when the view is replaced in the layout.

   Component-specific events, like Row Select Event Function, are also supported, and you can specify those in the properties grid of the view. Any function you name in these fields must be defined in `application-folder/src/scripts/ModuleName-ViewName/view-factory.js`. See *Kendo UI Builder by Progress: Modernizing OpenEdge Applications* for more information about extensions and event functions.

   The following sections describe how to edit each of the views described above.

Login view

To edit the log in view:

1. From the main editing screen, click **Edit** next to the **Application** module, at which point the login view will automatically open. If you're already in the module editing screen, select **login** from the list of views.
2. You can edit the following details of the log in screen:

   - **Custom HTML sections**: You can specify custom HTML code for the top, middle, and bottom of the login window. The Designer automatically generates three files for you in the folder *application-folder*`/src/html/Application-login`. Add your HTML code to the three files as desired.
   - **Header title**: The text that appears between the top and middle custom HTML sections.
   - **Logo**: An image that appears next to the header title. Enter the name of the file here, and place the file itself in *application-folder*`/src/assets/images`.
   - **Username label**: The text that appears above the box for app users to enter their usernames.
   - **Password label**: The text that appears above the box for app users to enter their passwords.
   - **Login Event Function**: The name of a function that runs when the user clicks the **Login** button. Any function you name in this field must be defined in *application-folder*`/src/scripts/Application-login/view-factory.js`.

Landing page view

To edit the landing page view:

1. From the main editing screen, click **Edit** next to the **Application** module, at which point the login view will automatically open. If you're already in the module editing screen, select **landing-page** from the list of views.
2. **Custom HTML sections**: You can edit the top and bottom HTML sections for the landing page, which appear above and below icons representing the application's modules. The Designer automatically generates two files for you in *application-folder*`/src/html/Application-landing-page`. Add your HTML code to the files as desired.

Data-Grid view

Selecting the Data-Grid view arranges the included fields from your data source by column, in the same order you arranged them in the **Included Fields** list in the data source. The Data-Grid has a number of customizable properties, listed on the right of the screen:

- **Grid Name**: A name for internal use only. The name that will appear for navigation on the main page of the app is the name you chose when first creating the view.
- **Grid Title**: The title that will appear when this view is actually on the screen in the app.
- **Data Provider**: The different data provider for this grid.
- **Data Source**: The desired data source from the chosen data provider.
- **Grid Columns**: Click **Edit** to change the included columns and various properties of the columns. See Editing grid columns on page 16 for more information.

- **Page Size**: The number of rows that will be displayed per page in the grid.
- **Row Template Function Name**: A JavaScript function name that returns the text for a Kendo UI row template. If specified, the row template is used to format all the rows in the grid. For more information, see the entry for rowTemplate at http://docs.telerik.com/kendo-ui/api/javascript/ui/grid#configuration-rowTemplate, in the *Kendo UI by Progress Documentation and API Reference*.
- **Row Template ID**: The ID of an HTML row template. If specified, the row template must be located in either the *application-folder*/src/html/*ModuleName-ViewName*/topSection.html file, the /middleSection.html file, or the /bottomSection.html file. If both a Row Template Function and a Row Template ID are indicated, the Row Template ID takes precedence.
- **Enable Column Filtering**: Provides the app user with the ability to filter the displayed information based on content, including filters for numeric values (e.g., is equal to, is not equal to) and text fields (e.g., starts with, does not contain, etc.).
- **Enable Grouping**: Provides the app user with the ability to group the rows according to the value of a particular column. For example, in a grid with columns representing customer names and order numbers, grouping by customer name will arrange the rows so that all of the order numbers from each given customer are grouped together. Grouping should only be used with client-side processing.
- **Enable Column Resize**: Provides the app user with the ability to resize the columns by dragging the column divider(s).
- **Enable Column Reordering**: Provides the app user with the ability to reorder the columns by dragging and dropping the column headers.
- **Enable Sorting**: Provides the app user with the ability to reorder rows by ascending or descending value in a given column.
- **Custom Sections**: You can add your own custom HTML code for three sections in the data-grid view:
  - **Top Section:** The area above the Grid Title.
  - **Middle Section**: The area below the Grid Title but above the grid itself.
  - **Bottom Section**: The area below the grid.

  The Designer automatically generates three files for you in the folder *application-folder*/src/html/*ModuleName-ViewName*. Add your custom HTML code to those files as desired.
- **Events**: Use **Row Select Event Function** to indicate a JavaScript function that will run when a grid row is selected by the app user. The code should be included in the file *application-folder*/src/html/*ModuleName-ViewName*/view-factory.js. See *Kendo UI Builder by Progress: Modernizing OpenEdge Applications* for more information about extensions and event functions.

See also

Adding data sources on page 11
Editing grid columns on page 16

Editing grid columns

From this dialog, you can change several properties of the columns. For example:

- You can use the **Include all** and **Exclude all** buttons to move the entire list of fields back and forth between **Included Fields** and **Excluded Fields** lists. You can also drag and drop single fields back and forth between the lists.
- When you select a particular field, you can:
  - Change the **Title Text**, which is the column heading users will see in the app.
  - **Enable HTML coding**: When checked, any HTML coding included in the field is applied. For example, with HTML coding enabled, a field containing "Jane" appears as "**Jane**" instead of normal text with the tags visible.
  - **Format**: This can be used for additional customization in how the field is presented. Use the argument {0} to represent the content of the column. For example, you can add additional text to be displayed. If a column holds a numeric value measured in kilograms, then:

    ```
    {0} kg
    ```

    appends " kg" to every field in the column. You can also add specifiers to control the format of the field. If a field is a date value, you can specify that a column should display dates in long form, e.g., Wednesday, October 3, 1997, using this syntax:

    ```
    {0:D}
    ```

    See http://docs.telerik.com/kendo-ui/framework/globalization/numberformatting and http://docs.telerik.com/kendo-ui/framework/globalization/dateformatting for more information on number and date formatting options, respectively.
  - **Template**: You can use this to apply any valid Kendo template to an entire column. For example, if you have a field named LastName, then entering:

    ```
    <strong>#: LastName #</strong>
    ```

    applies boldface to every data item in that column. See http://docs.telerik.com/kendo-ui/api/javascript/ui/grid#configuration-columns.template for more information.
  - Specify the initial **Width**, measured in pixels, of the column. Leaving this value empty allows the column width to be responsive as needed.

The following options are also available for the Grid component in the Blank view:

- **Enable Sorting**: Provides the app user with the ability to reorder rows by ascending or descending value in a given column.
- **Enable Filtering**: Provides the app user with the ability to filter the displayed information based on content, including filters for numeric values (e.g., is equal to, is not equal to) and text fields (e.g., starts with, does not contain, etc.).

Data-Grid-Form view

Selecting the Data-Grid-Form view creates a split screen. On the left, the form arranges the included fields from your data source by column, in the same order as they appear in the **Included Fields** list. On the right, single highlighted rows are presented in a form view. This view can be read-only or user editable.

The Data-Grid-Form has a number of customizable properties, listed on the right side of the screen:

- **Grid Name**: A name for internal use only. The name that will appear for navigation on the main page of the app is the name you chose when first creating the view.
- **Title**: The title that appears when this view is actually on the screen in the app.

- **New Title**: The title that appears when the app user adds a new record. (This is only relevant for an editable view.)
- **Edit Title**: The title that appears when the app user edits an existing record. (This is only relevant for an editable view.)
- **Confirm Delete**: When checked, the user is prompted with a confirmation dialog when deleting an existing record. (This is only relevant for an editable view.)
- **Data Provider**: The data provider for this grid.
- **Data Source**: The desired data source from the chosen data provider.
- **Edit Mode**: Choose **Read-Only** or **Read-Only to Edit**. Read-Only to Edit means that app users will be allowed to edit data in the form on the same screen. (The **Title** will change to **Edit Title** or **New Title**, depending on the type of edit the user is making.) The user's options for editing are determined by the default or overridden editor type, e.g., users will have access a basic editor with formatting capabilities for a rich text field.
- **Grid Columns**: Click **Edit** to change the included columns and various properties of the columns. See Editing grid columns on page 16 for more information.
- **Form Fields**: Click **Edit** to change the included fields and various properties of the form fields. See Editing form fields on page 19 for more information.
- **Page Size**: The number of rows that will be displayed per page in the grid.
- **Row Template Function Name**: A JavaScript function name that returns the text for a Kendo UI row template. If specified, the row template is used to format all the rows in the grid. For more information, see the entry for rowTemplate at http://docs.telerik.com/kendo-ui/api/javascript/ui/grid#configuration-rowTemplate, in the *Kendo UI by Progress Documentation and API Reference*.
- **Row Template ID**: The ID of an HTML row template. If specified, the row template must be located in either the `application-folder/src/html/ModuleName-ViewName/topSection.html` file, the `/middleSection.html` file, or the `/bottomSection.html` file. If both a Row Template Function and a Row Template ID are indicated, the Row Template ID takes precedence.
- **Enable Column Filtering**: Provides the app user with the ability to filter the displayed information based on content, including filters for numeric values (e.g., is equal to, is not equal to) and text fields (e.g., starts with, does not contain, etc.).

Note: Filtering, sorting, and paging operations execute based on the **Client-side processing** setting for the data source. Operations are performed on the client if **Client-side processing** is selected; otherwise, the operations are performed on the server side. See the info on enabling server-side processing in Adding data sources on page 11.

- **Enable Grouping**: Provides the app user with the ability to group the rows according to the value of a particular column. For example, in a grid with columns representing customer names and order numbers, grouping by customer name will arrange the rows so that all of the order numbers from each given customer are grouped together. Grouping should only be used with client-side processing.
- **Enable Column Resize**: Provides the app user with the ability to resize the columns by dragging the column divider(s).
- **Enable Column Reordering**: Provides the app user with the ability to reorder the columns by dragging and dropping the column headers.
- **Enable Sorting**: Provides the app user with the ability to reorder rows by ascending or descending value in a given column.

- **Custom Sections**: You can add your own custom HTML code for three sections in the data-grid-form view:
  - **Top Section:** The area above the Grid Title.
  - **Middle Section**: The area below the Grid Title but above the grid itself.
  - **Bottom Section**: The area below the grid.

  The Designer automatically generates three files for you in the folder *application-folder*/src/html/*ModuleName-ViewName*. Add your custom HTML code to those files as desired.
- **Events**: Use **Row Select Event Function** to indicate a JavaScript function that will run when a grid row is selected by the app user. The code should be included in the file *application-folder*/src/html/*ModuleName-ViewName*/view-factory.js. See *Kendo UI Builder by Progress: Modernizing OpenEdge Applications* for more information about extensions and event functions.

See also

Editing grid columns on page 16
Editing form fields on page 19

Editing form fields

From this dialog, you can change several properties of the form fields. For example:

- You can use the **Include all** and **Exclude all** buttons to move the entire list of fields back and forth between **Included Fields** and **Excluded Fields** lists. You can also drag and drop single fields back and forth between the lists.
- When you select a particular field, you can:
  - Specify the **Label Text** for the field.
  - **Format**: This can be used for additional customization in how the field is presented. Use the argument {0} to represent the content of the column. For example, you can add additional text to be displayed. If a column holds a numeric value measured in kilograms, then:

    ```
    {0} kg
    ```

    appends " kg" to every field in the column. You can also add specifiers to control the format of the field. If a field is a date value, you can specify that a column should display dates in long form, e.g., Wednesday, October 3, 1997, using this syntax:

    ```
    {0:D}
    ```

    See http://docs.telerik.com/kendo-ui/framework/globalization/numberformatting and http://docs.telerik.com/kendo-ui/framework/globalization/dateformatting for more information on number and date formatting options, respectively.

Data-Grid-Separate-Form view

Selecting the Data-Grid-Separate-Form view works in much the same way the Data-Gird-Form view works, except with this view, the user sees only the grid until they select a row. At that point, the view changes to the form, which you can make read-only or user editable, as with the Data-Grid-Form view. The Data-Grid-Separate-Form has a number of customizable properties, listed on the right side of the screen:

- **Grid Name**: A name for your internal use only. The name that will appear for navigation on the main page of the app is the name you chose when first creating the view.
- **Title**: The title that appears when this view is actually on the screen in the app.
- **New Title**: The title that appears when the app user adds a new record. (This is only relevant for an editable view.)
- **Edit Title**: The title that appears when the app user edits an existing record. (This is only relevant for an editable view.)
- **Confirm Remove**: When checked, the user is prompted with a confirmation dialog when deleting an existing record. (This is only relevant for an editable view.)
- **Data Provider**: The data provider for this grid.
- **Data Source**: The desired data source from the chosen data provider.
- **Edit Mode**: Choose **Read-Only** or **Read-Only to Edit**. Read-Only to Edit means that app users will be allowed to edit data in the form on the same screen. (The **Title** will change to **Edit Title** or **New Title**, depending on the type of edit the user is making.) The user's options for editing are determined by the underlying data type, e.g., users will have access a basic editor with formatting capabilities for a rich text field.
- **Grid Columns**: Click **Edit** to change various properties of the columns. See Editing grid columns on page 16 for more information.
- **Form Fields**: Click **Edit** to change various properties of the form fields. See Editing form fields on page 19 for more information.
- **Page Size**: The number of rows that will be displayed per page in the grid.
- **Row Template Function Name**: A JavaScript function name that returns the text for a Kendo UI row template. If specified, the row template is used to format all the rows in the grid. For more information, see the entry for rowTemplate at http://docs.telerik.com/kendo-ui/api/javascript/ui/grid#configuration-rowTemplate, in the *Kendo UI by Progress Documentation and API Reference*.
- **Row Template ID**: The ID of an HTML row template. If specified, the row template must be located in either the `application-folder/src/html/ModuleName-ViewName/topSection.html` file, the `/middleSection.html` file, or the `/bottomSection.html` file. If both a Row Template Function and a Row Template ID are indicated, the Row Template ID takes precedence.
- **Enable Column Filtering**: Provides the app user with the ability to filter the displayed information based on content, including filters for numeric values (e.g., is equal to, is not equal to) and text fields (e.g., starts with, does not contain, etc.).
- **Enable Grouping**: Provides the app user with the ability to group the rows according to the value of a particular column. For example, in a grid with columns representing customer names and order numbers, grouping by customer name will arrange the rows so that all of the order numbers from each given customer are grouped together. Grouping should only be used with client-side processing.
- **Enable Column Resize**: Provides the app user with the ability to resize the columns by dragging the column divider(s).
- **Enable Column Reordering**: Provides the app user with the ability to reorder the columns by dragging and dropping the column headers.
- **Enable Sorting**: Provides the app user with the ability to reorder rows by ascending or descending value in a given column.
- **Custom Sections**: You can add your own custom HTML code for three sections in the data-grid-separate-form view:

- **Top Section:** The area above the Grid Title.
- **Middle Section**: The area below the Grid Title but above the grid itself.
- **Bottom Section**: The area below the grid.

The Designer automatically generates three files for you in the folder *application-folder*/src/html/*ModuleName-ViewName*. Add your custom HTML code to those files as desired.

- **Events**: Use **Row Select Event Function** to indicate a JavaScript function that will run when a grid row is selected by the app user. The code should be included in the file *application-folder*/src/html/*ModuleName-ViewName*/view-factory.js. See *Kendo UI Builder by Progress: Modernizing OpenEdge Applications* for more information about extensions and event functions.

See also

Editing grid columns on page 16
Editing form fields on page 19

Blank view

Selecting the Blank view gives you the ability to define a custom layout and functionality of a view. The Blank view allows you to:

- configure one or more data source instances for the view
- drag-and-drop rows and columns to create a responsive layout
- drag-and-drop content components for navigation, scheduling, editing, and data management

See also

Adding data source instances for a blank view on page 21
Editing the layout on page 22
Adding content on page 23
Adding navigation on page 38

Adding data source instances for a blank view

From the **View Data Sources** dialog, you can review and add one or more data source instances to be used in your blank view. These instances are defined based on the data sources defined for your app but are scoped to a particular blank view. To open and use this dialog:

1. Click the **Edit** button next to **View Data Sources** at the top of the **View Properties** column.
2. Click **Add Data Item** to add a new data source instance from the data sources available for your app, or select an existing data source to review its details.
3. Enter a **Name** for the data source. This is the name you will select as a **Data Source** or **Model** when you want to associate a component with a data source. (*Models* will be explained in more detail in the section on Components.)
4. Select a **Data Provider** and **Data Source**. Any provider or source you created for your app from the main editing screen are selectable here.

5. Choose a **Page Size**. This number indicates how many records per page from your **Data Source** will be available to any component you associate the source with.
6. Click **Save** when done. Note that you can delete existing data sources by using the trashcan icons.

A data source is used with components that work with a set of records. From an ABL point of view, you can think of a data source as a temp-table. When a control works directly with a data source, the binding is two-way, and the JSDO data source and the UI data are always in sync. Often, the control can call read/save directly. Two controls are data-bound and use the data source directly:

- Grid
- List View

Note: All of the components and their properties are discussed in Adding content on page 23.

A *data model* is used to represent the values of a single record. The Kendo UI Builder automatically creates a data model for every data source instance created for a view. From an ABL point of view, you can think of a model for a data source instance as a buffer for a temp-table. When a component works with a data model, the binding is two-way, and the UI and the data are always in sync. However, there is NO association between the model and any JSDO data source. Moving values from the model to the data source must be done programmatically by the developer.

This simplifies the use case of the developer creating a form, either associated with a data-bound control or not. For example, to create a view with a grid and a form, the grid would be bound to the data source and the form fields would be “bound” to the data model. When a row is selected in the grid, the framework will automatically copy the values from the data source into the data model of the same name, so the currently selected record will be shown in the form. When you edit the form fields, the data model is updated; however, these changes are not automatically pushed back to the data source. The developer is responsible for this action and can use the helper dsService API. For more information on this API, see Kendo UI Builder by Progress: Modernizing OpenEdge Applications.

The controls that use a data model are:

- Editing components
- Combo Box
- Drop Down List

The combo box and drop down list are unique in that they have both a data source and a data model. The data source represents the list of available values, and the data model holds the selected item. You can think of a combo box or a drop down list as a drop down list (data source) combined with a text field (data model).

Editing the layout

The blank view supports a responsive layout for different screen sizes. Four screen sizes are supported for the blank view: Desktop, Laptop, Tablet Landscape and Tablet Portrait. The two basic layout elements in the blank view are rows and columns. You can drag and drop these elements from the **Components** palette:

- **Row**: Rows act as containers for columns. They have no additional properties of their own. You can move or delete existing rows, and you can also nest one or more rows inside a column if that column is either empty or already contains one or more rows.
- **Column**: Once a row has been added, you can drag and drop one or more columns into that row. You can delete columns and their contents by clicking on the thick top border and then clicking the trash can icon. You can also drag a column (and their contents) by its thick top border to different positions in the layout.

You can access a column's properties by clicking the thick border at the top of the column. The properties for a column are:

- **Screen widths**: You can specify, or change the width (**Column span**) of the column for each screen size. There are 12 "slots" available for columns in each row, and a column can take up one or more slots. For example, you can have 12 columns that span one slot each, or three columns that span four slots each.

  Note that the Tablet Landscape width is required and will be used for all screen sizes unless you enter a different width for the other screen sizes (Desktop, Laptop, and Tablet Portrait).

- **Hide on**: You can hide a column on a given screen size by selecting the checkbox for that screen width. You can use this feature to provide more details on larger screens while minimizing the content shown for smaller devices.

Use the display type preview at the top right of the toolbar to view the results of these changes.

Adding content

The Blank view offers a variety of data management, editing, scheduling, and navigational components. All of the components have an Id property:

- **Id**: A unique string that is automatically generated by the Designer for each component. You can use this to associate two components, like a label and a check-box, for example. Drag and drop a label and check-box, and then input the check-box's ID into the **For** property of the label to attach that label to the check-box. You can edit the Id to a more meaningful value if desired.

Like the predefined, data-driven views, the Blank view also provides the option for a custom section:

- **Custom Section**: You can add your own custom HTML code in this section. The Designer automatically generates a `topSection.html` file for you in the folder *`application-folder`*`/src/html/`*`ModuleName-ViewName`*. Add your custom HTML code to this file as desired. This file is typically used to add template code for the ***Template*** properties of content components added to the Blank view.

  Note: Unlike the custom section files generated for the predefined, data-driven views, none of the HTML content added to the the Blank view's `topSection.html` file is visible in the view.

Note: This topic covers data management, editing, and scheduling components. The navigation components are described in Adding navigation on page 38.

Table 1: Data management components

Component	Properties and notes
Grid	• **Data Source Name**: A data source instance that you've defined for the given blank view. • **Edit Mode**: Sets the user edit mode to none, inline editing, or editing in a pop-up window. • **Grid Columns**: Click **Edit** to change the included columns and various properties of the columns. See Editing grid columns on page 16 for more information.

Component	Properties and notes
	• **Row Template Function Name**: A JavaScript function name that returns the text for a Kendo UI row template. If specified, the row template is used to format all the rows in the grid. For more information, see the entry for rowTemplate at http://docs.telerik.com/kendo-ui/api/javascript/ui/grid#configuration-rowTemplate, in the *Kendo UI by Progress Documentation and API Reference*. • **Row Template ID**: The ID of an HTML row template. If specified, the row template must be located in the *application-folder*/src/html/*ModuleName-ViewName*/topSection.html file. If both a Row Template Function and a Row Template ID are indicated, the Row Template ID takes precedence. • **Enable Column Filtering**: Provides the app user with the ability to filter the displayed information based on content, including filters for numeric values (e.g., is equal to, is not equal to) and text fields (e.g., starts with, does not contain, etc.). • **Enable Grouping**: Provides the app user with the ability to group the rows according to the value of a particular column. For example, in a grid with columns representing customer names and order numbers, grouping by customer name will arrange the rows so that all of the order numbers from each given customer are grouped together. Grouping should only be used with client-side processing. • **Enable Column Resize**: Provides the app user with the ability to resize the columns by dragging the column divider(s). • **Enable Column Reordering**: Provides the app user with the ability to reorder the columns by dragging and dropping the column headers. • **Enable Sorting**: Provides the app user with the ability to reorder rows by ascending or descending value in a given column. • **Selection Type**: Determines whether a rows in a grid can be selected, and if so, if single or multiple selections are allowed. • **Model**: The grid component can be bound directly to a data source or associated with a data source through a model. For the latter, select the data source instance with this menu. • **Row Select Event Function**: An event that fires when a grid row is selected by the app user.Any function you name in this field must be defined in *application-folder*/src/html/*ModuleName-ViewName*/view-factory.js. • **Data Bound Event Function**: An event that fires after the user makes a change to one or more fields in the grid. Any function you name in this field

Component	Properties and notes
	must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
List View	• **Template Function**: The list view requires custom code to display data. See the code example below this table. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Template Id**: The Id of an HTML row template. If specified, the row template must be located in the *application-folder*/src/html/*ModuleName-ViewName*/topSection.html file. • **Edit Template Function**: A function used when the list view is in edit mode. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Edit Template Id**: The Id of an HTML row template when the list view is in edit mode. If specified, the row template must be located in the *application-folder*/src/html/*ModuleName-ViewName*/topSection.html file. • **Selection Type**: Determines whether a rows in a list view can be selected, and if so, if single or multiple selections are allowed. • **Select Event Function**: An event that fires after the user makes a selection in the view. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Cancel Event Function**: An event that fires after the user cancels an operation. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Data Bound Event Function**: An event that fires after the user makes a change to one or more fields in the grid. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Edit Event Function**: An event that fires after the user makes an edit in the view. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Remove Event Function**: An event that fires after the user removes a row from the view. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Save Event Function**: An event that fires after the user saves a change to the view. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Sample template function for list view

```
templateFunction: function(data) {
   var template = kendo.template("<div id='box'>#: Name #</div>");
   var result = template(data); //Passing the data to the template
   return result;}
```

Table 2: Editor components

Component	Properties, events, and notes
Check Box	• **Value**: Check the box to set the initial state of the check-box to checked, or leave it unchecked to leave the check-box unchecked. • **Disabled**: Check the box to disable the check-box, or leave it unchecked to make the check-box active. • **Model**: The data model associated with this component. • **Title Text**: This optional text appears when a user hovers over the component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Combo Box	This component binds to both a data source instance and a model. • **Data Source Name**: A data source instance that you've defined for the given blank view. The list of values for the combo box are loaded from this data source. • **Model**: The data model associated with this component. The selected value displayed in the combo box is loaded from this model. • **Data Text Field**: A field from the data source associated with the component. This is the field displayed in the combo box. • **Data Value Field**: A field from the data source that uniquely identifies the selected item. For example, employee records could have an Employee Number field that uniquely identifies employees with the same last name. • **Title Text**: This optional text appears when a user hovers over the component. • **Value Primitive**: Restricts the model to one field of the record. • **Filter Method**: If a user begins typing in the combo box, the choices available will be limited. For example, if you choose "Starts with," users can type "w" and the options will be restricted to only those values that begin with "w". • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Select Event Function**: An event that fires after the user makes a selection. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Filtering Event Function**: An event that fires after the user filters the results. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Currency Text Box	A specialized version of a text-box for currency. • **Decimals**: The number of decimal places displayed. • **Default Value**: The amount initially displayed in the text box. The user must edit or delete this amount as needed. • **Placeholder**: The amount that initially appears in the text box. Unlike Default value, this text automatically disappears when the user clicks on the text box • **Down** and **Up Arrow Text**: The text shown when the user hovers over the down and up arrows. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Model**: The data model associated with this component. • **Step**: The interval the entered value changes for each click of the down or up arrow. • **Title Text**: This optional text appears when a user hovers over the component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Date Picker	• **Min and Max Date**: The minimum (earliest) and maximum (latest) dates that can be displayed. • **Default Date**: The date initially displayed. • **Date Format**: The format the date is displayed in once the user has made a selection. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Date Time Picker	• **Min and Max Date**: The minimum (earliest) and maximum (latest) dates that can be displayed. • **Default Date**: The date and time initially displayed. • **Date Format**: The format for how the date and time are displayed. • **Time Interval**: The interval, in minutes, between selectable values in the time drop-down menu, e.g., a value of 30 gives users the choice of 1:00, 1:30, 2:00, etc. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Disabled Text Box	Use the disabled text box to display read-only text. • **Default Value**: The text initially displayed in the text box. The user must edit or delete this text as needed. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data source instance associated with this component.

Component	Properties, events, and notes
Drop Down List	This component binds to both a data source instance and a model. • **Data Source Name**: A data source that you've defined for the given blank view. • **Model**: The data model associated with this component. • **Data Text Field**: A field from the data source or model associated with the component. This is the field displayed in the drop down list. • **Data Value Field**: A field from the data source that uniquely identifies the selected item. For example, employee records could have an Employee Number field that uniquely identifies employees with the same last name. • **Title Text**: This optional text appears when a user hovers over the component. • **Value Primitive**: Restricts the model to one field of the record. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Select Event Function**: An event that fires after the user makes a selection. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Editor	A customizable HTML editor.
	• **Encoded**: Indicates whether the editor submits encoded HTML.
	• **Resizable**
	  • **Content**: If checked, the user can resize the editor.
	  • **Min** and **Max**: The smallest and largest size the user can choose for the size of the editor.
	  • **Toolbar**: When checked, the formatting tools are limited to one row at the top of the editor, with the tools that do not fit organized into a drop-down menu. When unchecked, the tools are organized into as many rows as necessary.
	• **Title Text**: This optional text appears when a user hovers over the component.
	• **Tools**: Select one or more formatting tools available to your users. Once selected, any tool can be removed by clicking the **X** next to its name.
	• **Model**: The data model associated with this component.
	• **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Email Text Box	A specialized version of a text-box for e-mail addresses. • **Debounce**: The delay, in milliseconds, between user input and the firing of the Change event function. • **Default Value**: The e-mail address initially displayed in the text box. The user must edit or delete this text as needed. • **Placeholder**: The e-mail address initially displayed in the text box. Unlike Default value, this text automatically disappears when the user clicks on the text box. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Integer Text Box	A specialized version of a text-box for integer values only. • **Default Value**: The integer initially displayed in the text box. The user must edit or delete this text as needed. • **Placeholder**: The integer initially displayed in the text box. Unlike Default value, this text automatically disappears when the user clicks on the text-box. • **Down** and **Up Arrow Text**: Decrement or increment the shown value by the number indicated by **Step**. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Step**: The interval the entered value changes for each click of the down or up arrow. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Label	• **For**: An Id of another component, which this label will be associated with. This is optional; for example, you can use a label as a user aid without tying it to a specific component. • **Text**: The actual text of the label.

Component	Properties, events, and notes
Masked Text Box	A specialized version of a text-box that supports an input mask. • **Mask**: Specifies the input mask. See http://docs.telerik.com/kendo-ui/api/javascript/ui/maskedtextbox#configuration-mask for the supported mask rules. • **Prompt Char**: The character displayed in the mask before the user has inputted anything. • **Default Value** (optional): The initial text displayed. The prompt char appears whenever the user deletes this text without replacing it with an accepted value. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Numeric Text Box	A specialized version of a text-box for numeric values only. • **Decimals**: The number of decimals displayed. • **Default Value**: The number initially displayed in the text box. The user must edit or delete this text as needed. • **Format**: The numeric format for the input. See http://docs.telerik.com/kendo-ui/api/javascript/ui/maskedtextbox#configuration-mask for the supported numeric types. • **Placeholder**: The number initially displayed in the text box. Unlike Default value, this text automatically disappears when the user clicks on the text-box. • **Down** and **Up Arrow Text**: Decrement or increment the shown value by the number indicated by **Step**. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Step**: The interval the entered value changes for each click of the down or up arrow. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Password Text Box	A specialized version of a text-box for passwords. • **Placeholder**: The text initially displayed in the text box. This text automatically disappears when the user clicks on the text box. • **Debounce**: The delay, in milliseconds, between user input and the firing of the Change event function. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Percent Text Box	A specialized version of a text-box for percent values only. Use this when the percent values in the database must be multiplied by 100 before appending the % symbol for proper display (e.g., the database value of 0.256 will be displayed as 25.6%). • **Decimals**: The number of decimal places displayed. • **Default Value**: The value that initially appears in the text box. The user must edit or delete this text as needed. • **Placeholder**: The value that initially appears in the text box. Unlike initial text, this text automatically disappears when the user clicks on the text-box. • **Down** and **Up Arrow Text**: Decrement or increment the shown value by the number indicated by **Step**. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Step**: The interval the entered value changes for each click of the down or up arrow. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Percent Value Text Box	A specialized version of a text-box for percent values only. Use this when the percent values in the database only need the addition of the % symbol for proper display (e.g., the database value of 25 will be displayed as 25%). • **Decimals**: The number of decimal places displayed. • **Default Value**: The value that initially appears in the text box. The user must edit or delete this text as needed. • **Placeholder**: The value that initially appears in the text box. Unlike initial text, this text automatically disappears when the user clicks on the text-box. • **Down** and **Up Arrow Text**: Decrement or increment the shown value by the number indicated by **Step**. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Step**: The interval the entered value changes for each click of the down or up arrow. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Phone Text Box	A specialized version of a text-box for phone numbers. • **Mask**: Specifies the input mask. For example, (000) 000-0000 allows the user to enter a standard U.S., 10-digit phone number. • **Default Value** (optional): The initial text displayed. The prompt char appears whenever the user deletes this text without replacing it with an accepted value. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Radio Button List	A radio button set with two possible values. • **Default Value**: Sets the initial state of the buttons. Check the box to set the True button, or leave unchecked to set the False button. • **True and False text**: The labels that appear next to the respective radio buttons. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Slider	A rich input component for selecting numeric values. • **Min** and **Max**: The minimum and maximum values allowed, respectively. • **Orientation**: Choose a horizontal or vertical orientation for the slider. • **Show Buttons**: Select to display decrease and increase arrow buttons on either end of the slider. • **Step**: The interval that determines which values on the slider are valid, e.g., a step of 0.5 on a range of 1 to 10 allows users to select 1, 1.5, 2, 2.5, etc. • **Default Value**: The number at which the slider is initially positioned. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component using the keyboard, the buttons, or the drag handle. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js. • **Slide Event Function**: An event that fires after the user moves the drag handle. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Text Area	A multi-line version of a text box. • **Default Text**: The text that initially appears in the text box. The user must edit or delete this text as needed. • **Placeholder**: Text that initially appears in the text box. Unlike default text, this text automatically disappears when the user clicks on the text box. • **Rows**: The number of lines of text visible in the text box. • **Columns**: The width of the text box measured in average character widths. • **Debounce**: The delay, in milliseconds, between user input and the firing of the Change event function. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Text Box	• **Default Text**: The text that initially appears in the text box. The user must edit or delete this text as needed. • **Placeholder**: Text that initially appears in the text box. Unlike initial text, this text automatically disappears when the user clicks on the text box. • **Debounce**: The delay, in milliseconds, between user input and the firing of the Change event function. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data source instance associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Component	Properties, events, and notes
Time Picker	Allows users to select time values from a predefined list or enter new ones. • **Default Time**: The time initially displayed. • **Time Format**: The format for how the time is displayed. • **Time Interval**: The interval, in minutes, between selectable values in the time drop-down menu, e.g., a value of 30 gives users the choice of 1:00, 1:30, 2:00, etc. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
Url Text Box	A specialized version of a text-box for URLs. • **Debounce**: The delay, in milliseconds, between user input and the firing of the Change event function. • **Default Value**: The URL initially displayed in the text box. The user must edit or delete this text as needed. • **Placeholder**: The URL initially displayed in the text box. Unlike Default value, this text automatically disappears when the user clicks on the text box. • **Title Text**: This optional text appears when a user hovers over the component. • **Model**: The data model associated with this component. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Table 3: Scheduling components

Component	Properties and notes
Calendar	A graphical calendar that supports navigation and selection. • **Min and Max Date** (optional): The minimum (earliest) and maximum (latest) dates that can be displayed. • **Default Date** (optional): The date initially displayed. • **Date Format**: Specifies the date format used by the calendar component's underlying `value( )` function. See http://docs.telerik.com/kendo-ui/api/javascript/ui/calendar#configuration-value for more information. • **Change Event Function**: An event that fires after a user makes a change to the state of the component. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Table 4: Custom components

Component	Properties and notes
Custom HTML	A custom component that can contain valid HTML. • **HTML**: Text with desired HTML tags applied.

Adding navigation

You can add standalone buttons, or add a toolbar made up of buttons, separators, and templates in a toolbar.

Table 5: Navigation components

Component	Properties and notes
Button	A styled clickable UI with keyboard operability for elements. • **Text**: The text that appears on the button itself. • **Primary**: Indicates whether the button will be highlighted with a background color. • **Click Event Function**: An event that fires after a user clicks the button. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.

Toolbar	Holds different types of controls, such as buttons, button groups, toggle buttons, split buttons, and other customized elements.
	• **Toolbar Items**: Click **Edit** to add or remove items on the toolbar. Once in the **Toolbar Items** dialog, you can add or remove buttons, separators, and templates with the following properties.
	  • **Item ID**: Add a name for the toolbar item.
	  • **Item Type**: Select from button, separator, and template.
	  • **Text**: The text that appears on the item.
	  • **Togglable**: Indicates whether the item can be toggled on and off.
	  • **Template ID**: The Id of an HTML template for the toolbar wrapper. If specified, the template must be located in the *application-folder*/src/html/*ModuleName-ViewName*/topSection.html file.
	  • **Template Function**: A function used to define the toolbar wrapper. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
	• **Resizable**: If the size of the browser window changes in a way that hides one or more tools, this allows the toolbar to move the overflow into a drop down list.
	• **Click Event Function**: An event that fires after the user clicks the button. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.
	• **Toggle Event Function**: An event that fires after the user toggles a button. Any function you name in this field must be defined in *application-folder*/src/scripts/*ModuleName-ViewName*/view-factory.js.