Pagedegarde

sommaire

ControlsFX

[ControlsFX](https://controlsfx.github.io/)

 <https://controlsfx.github.io/>

* Features▾
  + [Actions](https://controlsfx.github.io/features/actions/)
  + [Borders](https://controlsfx.github.io/features/borders/)
  + [BreadCrumbBar](https://controlsfx.github.io/features/breadcrumbar/)
  + [CheckComboBox](https://controlsfx.github.io/features/checkcombobox/)
  + [CheckListView](https://controlsfx.github.io/features/checklistview/)
  + [CheckTreeView](https://controlsfx.github.io/features/checktreeview/)
  + [Decoration / Validation](https://controlsfx.github.io/features/decorationvalidation/)
  + [Dialog ‘Command Links’](https://controlsfx.github.io/features/dialog/)
  + [Glyph Font](https://controlsfx.github.io/features/glyph/)
  + [GridView](https://controlsfx.github.io/features/gridview/)
  + [HiddenSidesPane](https://controlsfx.github.io/features/hiddensidespane/)
  + [HyperlinkLabel](https://controlsfx.github.io/features/hyperlinklabel/)
  + [InfoOverlay](https://controlsfx.github.io/features/infooverlay/)
  + [ListSelectionView](https://controlsfx.github.io/features/listselectionview/)
  + [MasterDetailPane](https://controlsfx.github.io/features/masterdetailpane/)
  + [NotificationPane](https://controlsfx.github.io/features/notificationspane/)
  + [Notifications](https://controlsfx.github.io/features/notifications/)
  + [PlusMinusSlider](https://controlsfx.github.io/features/plusminusslider/)
  + [PopOver](https://controlsfx.github.io/features/popover/)
  + [PropertySheet](https://controlsfx.github.io/features/propertysheet/)
  + [RangeSlider](https://controlsfx.github.io/features/rangeslider/)
  + [Rating](https://controlsfx.github.io/features/rating/)
  + [SearchableComboBox](https://controlsfx.github.io/features/searchablecombobox/)
  + [SegmentedBar](https://controlsfx.github.io/features/segmentedbar/)
  + [SegmentedButton](https://controlsfx.github.io/features/segmentedbutton/)
  + [SnapshotView](https://controlsfx.github.io/features/snapshotview/)
  + [SpreadsheetView](https://controlsfx.github.io/features/spreadsheetview/)
  + [TaskProgressView](https://controlsfx.github.io/features/taskprogressview/)
  + [TextFields](https://controlsfx.github.io/features/textfields/)
  + [ToggleSwitch](https://controlsfx.github.io/features/toggleswitch/)
  + [World Map View](https://controlsfx.github.io/features/worldmapview/)
* [Javadocs](https://controlsfx.github.io/javadocs/)
* [Translations](https://controlsfx.github.io/translations/)
  + [ControlsFX Transifex](https://controlsfx.github.io/translations/transifex/)
* [FXSampler](https://controlsfx.github.io/fxsampler/)

# **Actions API**[**#**](https://controlsfx.github.io/features/actions/#actions-api)

The [Actions](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/action/Action.html) API is used by both the dialogs API and the ButtonBar API. It essentially abstracts away the notion of how a control is represented, instead focusing on the properties of the control (e.g. text, graphic, etc). This makes it really easy to reuse the same Action in multiple places in your UI. We’ve also developed convenience API to convert Actions into the common UI controls (see the ActionUtils class for these methods). In recent releases, we’ve built on to the Actions API to include functionality such as:

* An [ActionProxy annotation](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/action/ActionProxy.html) (to create actions directly by annotating methods) which can be discovered at runtime using the [ActionMap](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/action/ActionMap.html) class.
* [ActionTextBehavior](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/action/ActionUtils.ActionTextBehavior.html) to more easily specify what text is shown in UI controls when built from an action.

# **Borders**[**#**](https://controlsfx.github.io/features/borders/#borders)

The Borders API is a simple (and still relatively exploratory) API that allows you to easily wrap panes or nodes with a border, similar to the Swing BorderFactory API. Here’s a some code, and then a screenshot of what that code results in:

Node wrappedButton = Borders.wrap(button)

.lineBorder().color(Color.RED).build()

.lineBorder().color(Color.GREEN).build()

.build();

A single button with two borders

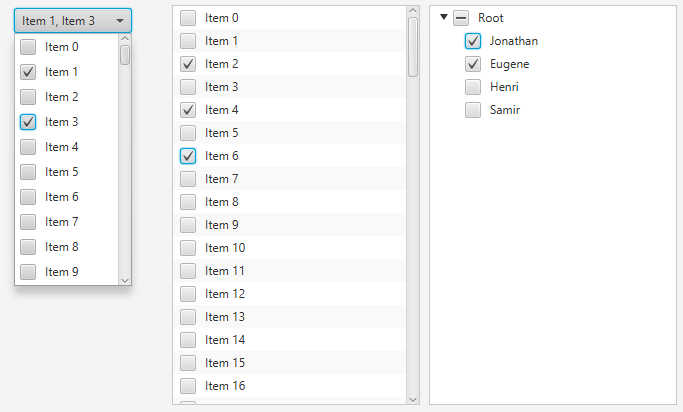
# **BreadCrumbBar**[**#**](https://controlsfx.github.io/features/breadcrumbar/#breadcrumbbar)

The BreadCrumbBar control is designed to allow people to easily navigate back up a hierarchical page structure. It’s based on the same API as the TreeView control (that is, it uses the TreeItem structure). This control is useful to visualize and navigate a hierarchical path structure, such as file system. Here’s what it currently looks like:

The BreadCrumbBar UI controlThe BreadCrumbBar UI control

# **CheckComboBox / CheckListView / CheckTreeView**[**#**](https://controlsfx.github.io/features/checkcombobox/#checkcombobox--checklistview--checktreeview)

All three controls offer the same functionality – support for checking zero or more items in a ComboBox, ListView, or a TreeView, and being able to easily retrieve the checked items from an underlying ‘check model’. These were already easily doable using standard JavaFX APIs, but this takes it a step further by automatically tracking the checkbox selections for you. Here’s a screenshot of each of the new controls:

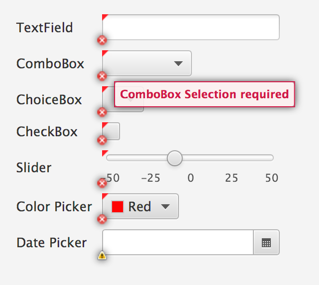
CheckComboBox, CheckListView, and CheckTreeView

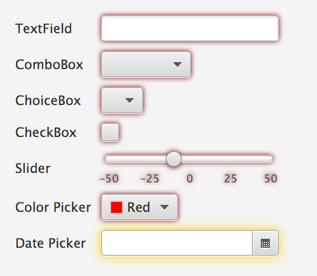
# **Decoration / Validation**[**#**](https://controlsfx.github.io/features/decorationvalidation/#decoration--validation)

ControlsFX has a decoration framework to allow developers to apply decorations to nodes on screen. For example, you may want to show graphics in the corner of your nodes, or add a styleclass in certain circumstances. The decoration framework makes this kind of action trivial. However, to be fair, the decoration framework was primarily developed to be the visual feedback mechanism for a validation framework, and we have one of those in ControlsFX too!

The validation framework is designed to be lightweight, but will grow over time as developers suggest desired functionality (we are definitely open to suggestions!).

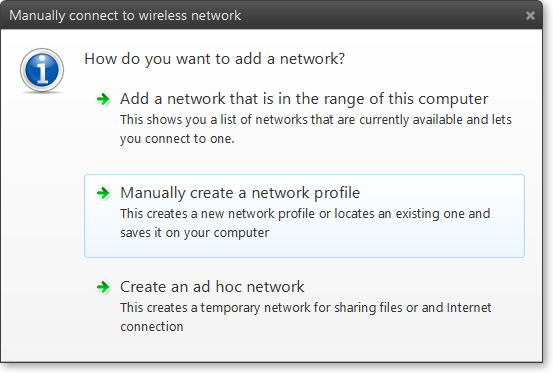
Here are two screenshots:

Graphic decorations showing how validation failed

CSS decorations showing how validation failed

# **Dialog ‘Command Links’**[**#**](https://controlsfx.github.io/features/dialog/#dialog-command-links)

Command links are common on the Windows platform. The version we have in ControlsFX looks like the following:

Dialogs Command Links

# **Glyph font**[**#**](https://controlsfx.github.io/features/glyph/#glyph-font)

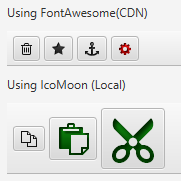
ControlsFX supports font packs. The following font packs:

* [Font Awesome](http://fontawesome.github.io/Font-Awesome/)
* [IcoMoon](http://icomoon.io/)

This support is not only in the form of API to render glyphs as graphics in Buttons, etc – it is also baked into our other API. For example, the new @ActionProxy API can have its graphic set from a font pack in the following manner:

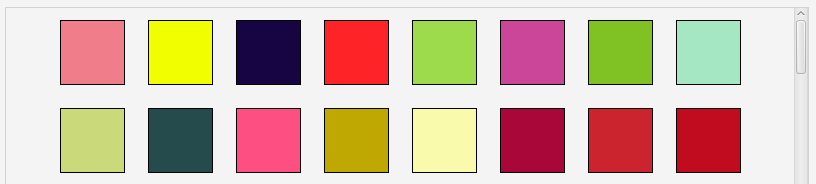
@ActionProxy(text=”Action Text”, image=”font>FontAwesome:STAR”)

This feature was inspired by **Jens Deters’** [FontAwesomeFX](https://bitbucket.org/Jerady/fontawesomefx).

GlyphFont

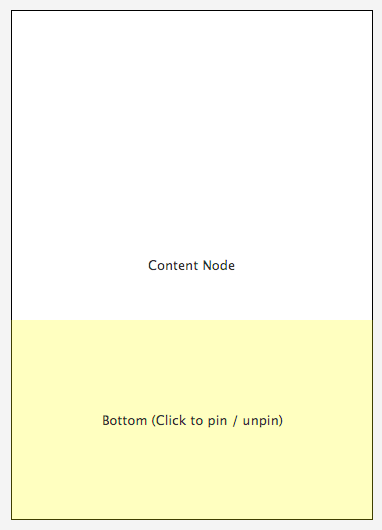
# **GridView**[**#**](https://controlsfx.github.io/features/gridview/#gridview)

[GridView](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/GridView.html) is a fast, virtualised grid container similar to ListView, except that there can be multiple cells on each row. Great for showing images!

GridView

# **HiddenSidesPane**[**#**](https://controlsfx.github.io/features/hiddensidespane/#hiddensidespane)

A pane used to display a full-size content node and four initially hidden nodes on the four sides. The hidden nodes can be made visible by moving the mouse cursor to the edges of the pane. The hidden node will appear (at its preferred width or height) with a short slide-in animation. The node will disappear again as soon as the mouse cursor exits it. A hidden node / side can also be pinned, and it will remain visible as long as it stays pinned.

The HiddenSidesPane, with a node at the bottom that has animated into view

# **HyperlinkLabel**[**#**](https://controlsfx.github.io/features/hyperlinklabel/#hyperlinklabel)

This control takes a text string (like a Label), but you can annotate that string with text in [square braces]. By doing this, the text in square braces is automatically rendered as a hyperlink. You can then register a callback to be notified when a user clicks on the hyperlinks in the text (and then do the correct thing, based on the selected hyperlink).

Essentially, the use case is when you have a “[block of text](http://www.github.com/controlsfx/controlsfx) in [which you](http://www.github.com/controlsfx/controlsfx/wiki) want to [embed hyperlinks](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/)” (kind of like what I just craftily did right there). In this situation you are left to painfully merge together Label and Hyperlink nodes, and hope everything works out. To save you this hassle, take HyperlinkLabel for a spin!

Here’s a thrilling screenshot of the control in action:

HyperlinkLabelHyperlinkLabel

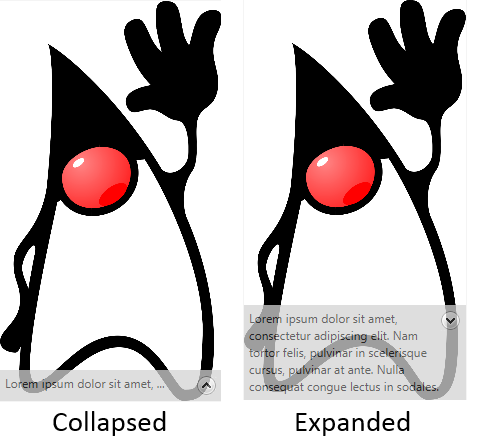
# **InfoOverlay**[**#**](https://controlsfx.github.io/features/infooverlay/#infooverlay)

This control was initially developed by **David Grieve** before being integrated into ControlsFX.

It is designed to show a small blurb of text above a node (most commonly an ImageView, but it will work with any Node). The text can be collapsed down to a single line, or expanded to show the entire text.

In some ways, it can be thought of as an always visible tooltip. By default, it is collapsed so only the first line is shown – hovering over it (or clicking on it if the showOnHover functionality is disabled) will expand it to show all text.

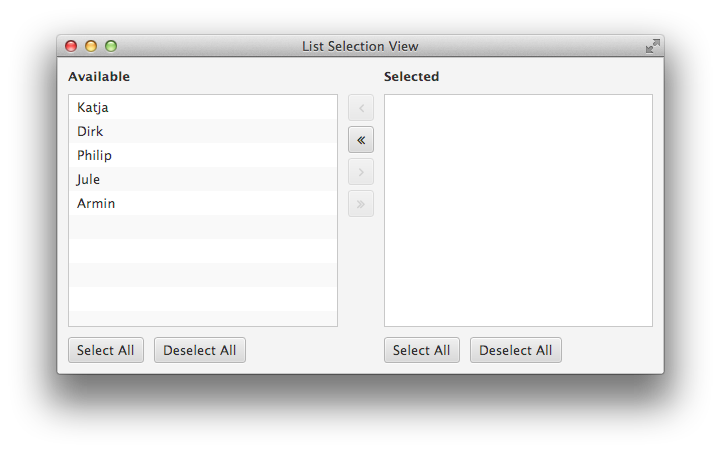
Shown below is a screenshot of the InfoOverlay control in both its collapsed and expanded states:

The InfoOverlay control

# **ListSelectionView**[**#**](https://controlsfx.github.io/features/listselectionview/#listselectionview)

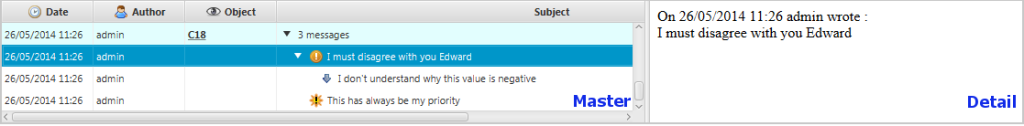
A control used to perform a multi-selection via the help of two list views. Items can be moved from one list (source) to the other (target). This can be done by either double-clicking on the list items or by using one of the “move” buttons between the two lists.

Each list can be decorated with a header and a footer node. The default header nodes are simply two labels: “Available” and “Selected”.

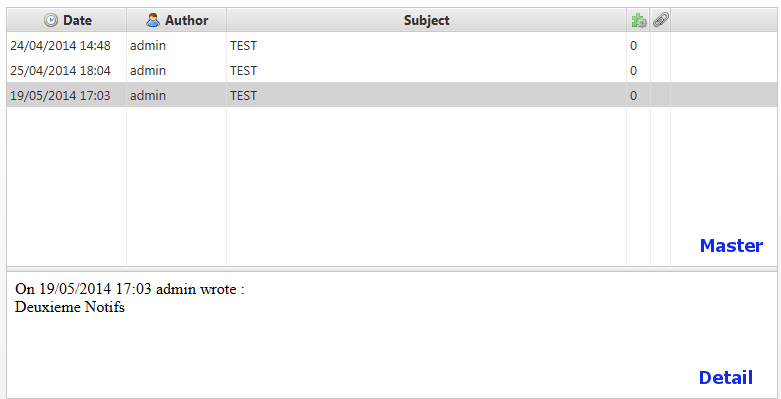
ListSelectionView control

# **MasterDetailPane**[**#**](https://controlsfx.github.io/features/masterdetailpane/#masterdetailpane)

A master / detail pane is used to display two nodes with a strong relationship to each other. Most of the time the user works with the information displayed in the master node but every once in a while additional information is required and can be made visible via the detail node. By default the detail appears with a short slide-in animation and disappears with a slide-out. This control allows the detail node to be positioned in four different locations (top, bottom, left, or right). Here’s a screenshot with a TableView in the ‘master’ section, and a text display in the ‘detail’ section.

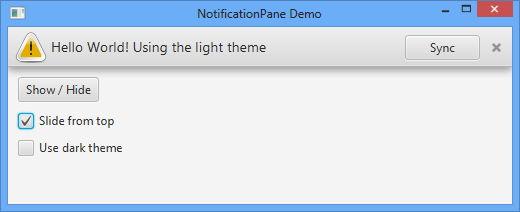
MasterDetailPane with DetailSide set to RIGHT

Here’s another screenshot, showing the master section above the detail section:

MasterDetailPane with DetailSide set to BOTTOM

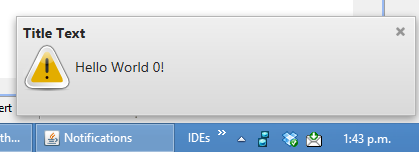
# **NotificationPane**[**#**](https://controlsfx.github.io/features/notificationspane/#notificationpane)

The [NotificationPane](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/NotificationPane.html) control allows you to notify your users of something without requiring their immediate input. The NotificationPane will animate in and out of view, and provides support for an arbitrarily complex ‘graphic’ property (put any Node in there), as well as text, and an area to place control buttons. It also supports two visual styles, continuing in our theme of providing light and dark styles.

NotificationPane

# **Notifications**[**#**](https://controlsfx.github.io/features/notifications/#notifications)

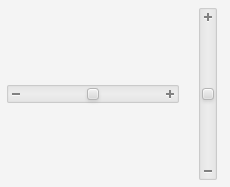
ControlsFX has had the NotificationPane feature for a release or two now, but that required that the notification appear within the window. Some use cases are to simply inform the user of an event, even when the main window itself is not visible (or focused) – think email notifications, for example. This is where the Notifications class comes in – it will show a notification message to users in one of nine locations on the screen, and even supports stacking multiple notifications, and showing complex scenegraphs. After a set duration, the notification will fade out.

Notifications

# **PlusMinusSlider**[**#**](https://controlsfx.github.io/features/plusminusslider/#plusminusslider)

A plus minus slider allows the user to continuously fire an event carrying a value between -1 and +1 by moving a thumb from its center position to the left or right (or top and bottom) edge of the control. The thumb will automatically center itself again on the zero position when the user lets go of the mouse button.

Scrolling through a large list of items at different speeds is one possible use case for a control like this. You may be familiar with Google Picasa, which has a similar control for scrolling through the image list.

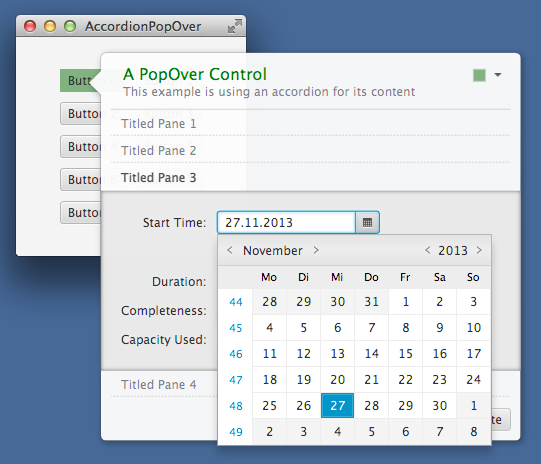
The PlusMinusSlider in both horizontal and vertical states

# **PopOver**[**#**](https://controlsfx.github.io/features/popover/#popover)

The PopOver control is designed to provide information to users in a visually pleasing way that does not interfere with their workflow. It is hard to explain, but hopefully the screenshot below better summarises the functionality:

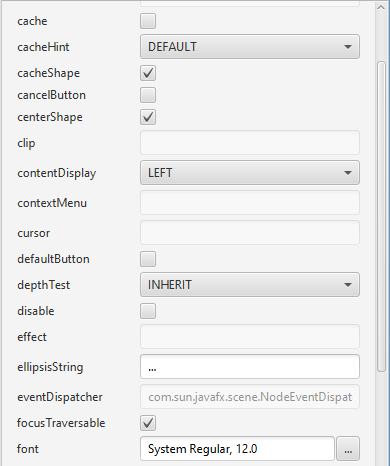
A PopOver with no content

Of course, a PopOver with no content is a little sad, so here’s a screenshot of one with content inside it. Note that this is using a custom TitledPane style that presently doesn’t ship with ControlsFX (but that you can create with a little CSS):

A PopOver with content

# **PropertySheet**[**#**](https://controlsfx.github.io/features/propertysheet/#propertysheet)

[PropertySheet](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/PropertySheet.html) is a vertical list of properties that can be edited with a number of pre-build editors. We support string, number, boolean, enumeration, color, and font editors. As well as supporting the built-in editors, you can provide custom editors for your specific use cases.

A PropertySheet Control

# **RangeSlider**[**#**](https://controlsfx.github.io/features/rangeslider/#rangeslider)

When you want a user to specify a range, right now you need to give them two text fields to type into (or two sliders). Yuck!

The [RangeSlider](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/RangeSlider.html) control is essentially the standard JavaFX slider but with two draggable thumbs on it – one representing the low value and the other the high value. Much better!

RangeSliderA RangeSlider Control

# **Rating**[**#**](https://controlsfx.github.io/features/rating/#rating)

[Rating](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/Rating.html) is a simple control that allows users to specify a rating between zero, and a maximum value. This control supports partial ratings and updating on a hover or on a click.

RatingA Rating Control

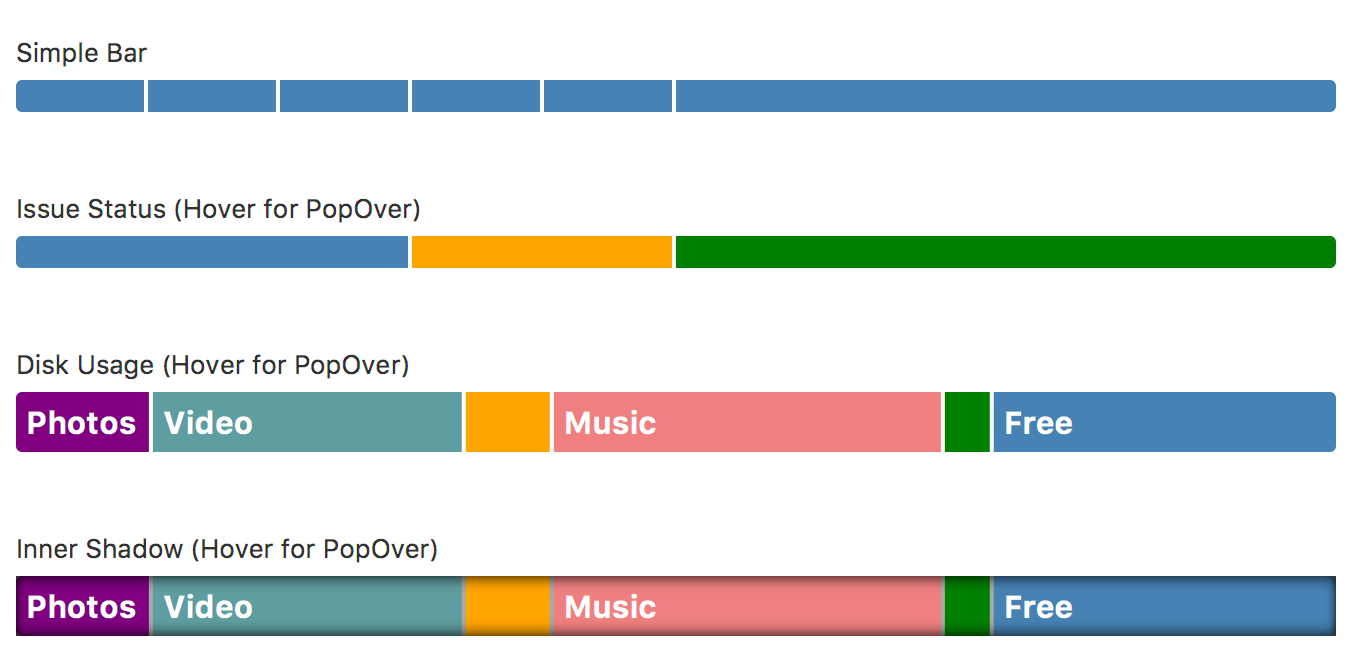
# **SearchableComboBox**[**#**](https://controlsfx.github.io/features/searchablecombobox/#searchablecombobox)

[SearchableComboBox](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/SearchableComboBox.html) is a simple extension of a ComboBox which shows a search field while the popup is showing. The user can type any text into this search field to filter the popup list.

A SearchableComboBox Control

# **SegmentedBar**[**#**](https://controlsfx.github.io/features/segmentedbar/#segmentedbar)

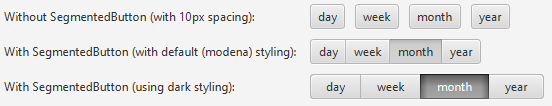
A control that makes it easy to create a horizontal or vertical bar that visualizes the segmentation of a total value. It consists of several segments, each segment representing a value. The sum of all values is the total value of the bar.

Segmented Bar Variations

# **SegmentedButton**[**#**](https://controlsfx.github.io/features/segmentedbutton/#segmentedbutton)

When you have a ToggleGroup with buttons in it, it looks nicer when the buttons are grouped together (rather than individual buttons).

This is exactly what the [SegmentedButton](https://controlsfx.github.io/javadoc/11.0.3/org.controlsfx.controls/org/controlsfx/control/SegmentedButton.html) control does - it ‘merges’ together all of your buttons in a ToggleGroup so that it is clear to users that the buttons are related.

A SegmentedButton Control

# **SnapshotView**[**#**](https://controlsfx.github.io/features/snapshotview/#snapshotview)

The SnapshotView control allows users to draw a bounding box inside a scenegraph (or typically in an ImageView).

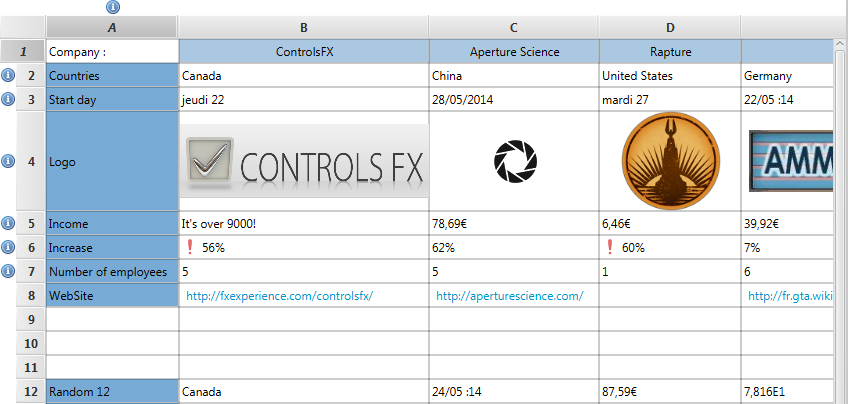
This bounding box represents the dimensions of their selection, and from which an image can be created of the selected area. This is another of those controls that is easier to describe with an image, so, see below! 🙂

SnapshotView - the 'Controls' section of the image has been selected

# **SpreadsheetView**[**#**](https://controlsfx.github.io/features/spreadsheetview/#spreadsheetview)

The SpreadsheetView control is exactly what you expect – a TableView-like control that supports cell spanning, fixed rows and columns, and cell editing.

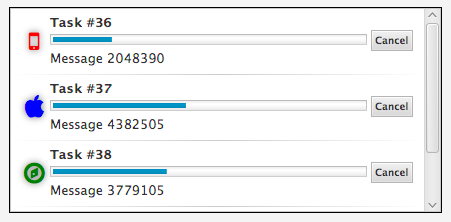
Here’s a screenshot of the control:

SpreadsheetView

# **TaskProgressView**[**#**](https://controlsfx.github.io/features/taskprogressview/#taskprogressview)

The task progress view is used to visualize the progress of long-running tasks. These tasks are created via the [Task](https://openjfx.io/javadoc/11/javafx.graphics/javafx/concurrent/Task.html) class. This view manages a list of such tasks and displays each one of them with their name, progress, and update messages.

An optional graphic factory can be set to place a graphic in each row. This allows the user to more easily distinguish between different types of tasks.

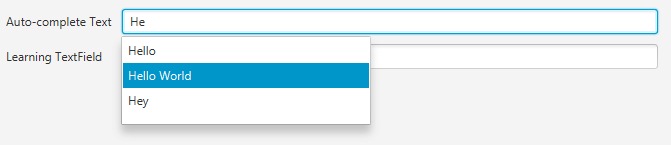
TaskProgressView

# **TextFields**[**#**](https://controlsfx.github.io/features/textfields/#textfields)

ControlsFX ships with a few useful TextField-related controls.

## AutoComplete TextField [#](https://controlsfx.github.io/features/textfields/#autocomplete-textfield)

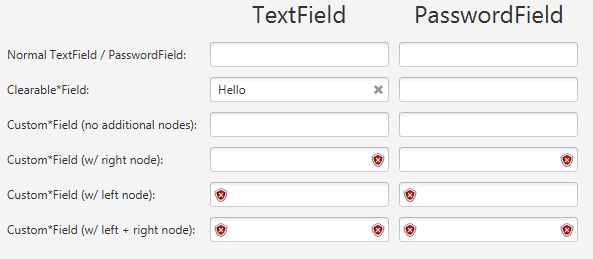
It can provide suggestions to users as they type into it. The popup will only show options based on the currently typed input.

AutoComplete TextField

## TextFields [#](https://controlsfx.github.io/features/textfields/#textfields-1)

In addition, there is API to make it really easy for you to customise a TextField (or PasswordField) by placing nodes within the text area. There are also pre-built features such as clearable fields that show a little cross that clears the text when deleted.

Here’s a screenshot:

TextField with nodes

# **ToggleSwitch**[**#**](https://controlsfx.github.io/features/toggleswitch/#toggleswitch)

Much like a Toggle Button this control allows the user to toggle between one of two states. It has been popularized in touch based devices where its usage is particularly useful because unlike a checkbox the finger touch of a user doesn’t obscure the control.

Toggle Switch

# **World Map View**[**#**](https://controlsfx.github.io/features/worldmapview/#world-map-view)

A simple map view showing either the entire world or a list of countries. The view is not capable of displaying detailed map information. The view is based on simple SVG data found in a properties file of the ControlsFX distribution.

A big advantage of this approach is the fact that the view can be run without a network connection. In addition to showing countries or the world the view can also show locations. The map can be customized by specifying custom factories for the country and location views.

World Map View

# **Javadocs**[**#**](https://controlsfx.github.io/javadocs/#javadocs)

One of the goals of ControlsFX is to have high quality JavaDocs, so one of the main drivers for every release is to improve those further.

As always, [the JavaDocs can be read online](https://controlsfx.github.io/javadoc) at <https://controlsfx.github.io/javadoc> , but they are also available as part of the download. The intention is that the JavaDocs should be the primary way of people learning how to use the ControlsFX API, so there are good code samples and screenshots of the controls in action.

Hopefully this will help to get people started quickly.

autre