列表和其他数据模型

应用程序通常需要处理和显示组织到列表或网格视图中的数据。模型、视图和委托用于此目的。它们将数据可视化模块化，以便您控制数据的不同方面。例如，可以将列表视图与网格视图交换，而对数据进行少量更改。同样，将数据实例封装在委托中允许开发人员指定如何呈现或处理数据。

*模型*包含数据及其结构。有几个组件可用于创建不同类型的模型。*视图*是在列表或网格中或沿路径显示数据的容器。*委托*人指示数据在视图中的显示方式。委托获取模型中的每条数据并将其封装。然后，可通过委托访问数据。

为了可视化数据，视图的 model 属性绑定到模型，委托属性绑定到组件。

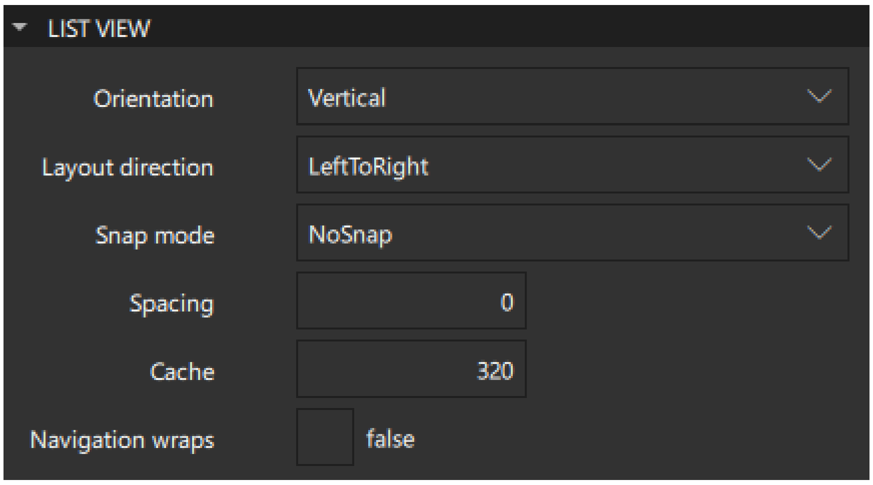
有关更多信息，请参见 Qt Quick 中的模型和视图。

列表和网格视图

创建**列表视图**和**网格视图**组件的实例，以列表或网格格式组织其他组件实例。它们在"**组件**>**默认组件**>**视图中**可用。

列表视图将其他组件组织为列表，而**网格视图**将它们组织为网格。默认情况下，列表视图和网格视图中的组件从左到右垂直排列。它们从左到右水平布局，从上到下垂直布局。

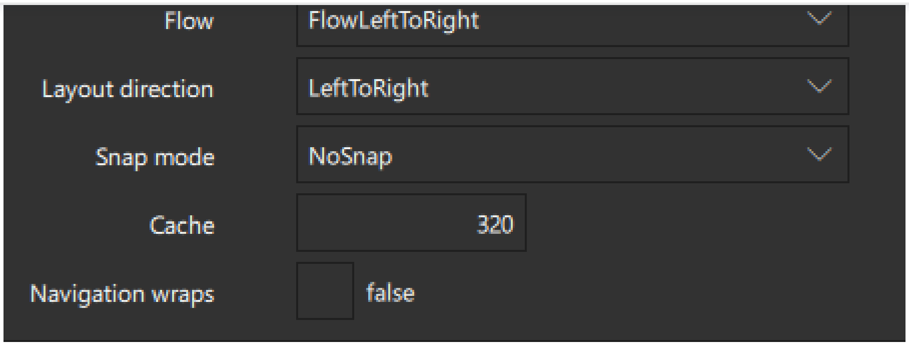
您可以在"**方向**"字段中更改列表视图方向，在"流动"字段中更改网格视图**流**。您可以在"布局方向"字段中更改**布局方向**。通过设置这些属性的值，可以生成各种布局。


LIST VIEW

Orientation	Vertical
Layout direction	LeftToRight
Snap mode	NoSnap
Spacing	0
Cache	320
Navigation wraps	false

对于列表视图，可以在"**间距**"字段中指定列表项之间的间距。对于网格视图，可以在 W 和 H 字段中指定每个单元格的宽度和高度。

GRID VIEW



Flow	FlowLeftToRight
Layout direction	LeftToRight
Snap mode	NoSnap
Cache	320
Navigation wraps	false

选中"**导航换行**"复选框以指定键导航在到达视图末尾时环绕并将所选内容移动到视图另一端的下一行或单元格。

列表视图和网格视图本质上都是可轻拂的。

"**捕捉模式**"字段的值决定了拖动或轻拂后视图滚动的稳定方式。默认情况下，视图停止在可见区域内的任意位置。如果选择"SnapToRow"，则视图将以与视图开头对齐的行（或网格视图自上而下的列）进行结算。如果选择"SnapOneRow"，则在释放鼠标按钮时，视图将距离第一个可见行不超过一行或一列。此选项对于一次移动一页特别有用。

委派缓存

"**缓存**"字段的值确定委托是否保留在视图的可见区域之外。

如果此值大于零，则视图可以保留实例化的任意数量的委托，以适合指定的缓存。例如，如果在垂直视图中，委托为 20 像素高，则有三列，缓存设置为 40，则可以创建或保留可见区域上方和下方的最多六个委托。缓存的委托是异步创建的，允许跨多个帧进行创建，并降低跳过帧的可能性。为了提高绘画性能，不会绘制可见区域外的委托。

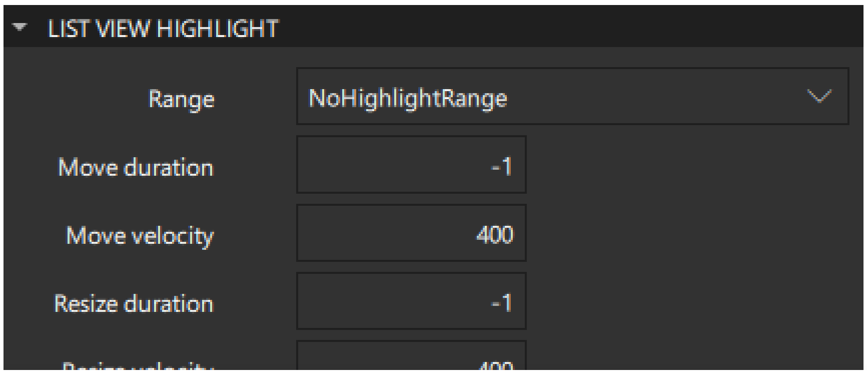
此属性的默认值取决于平台，但通常为大于零的值。负值将被忽略。

缓存不是像素缓冲区。它仅维护其他实例化的委托。

注意： 设置 Cache 属性不能替代创建高效委托。它可以提高滚动行为的平滑度，但代价是额外的内存使用量。委托中的项和绑定越少，视图的滚动速度就越快。重要的是要意识到，设置缓存只会推迟由加载缓慢的委托引起的问题，这不是这个问题的解决方案。

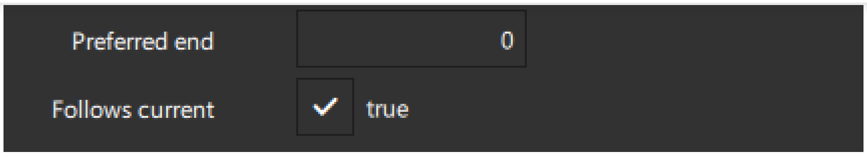
View Highlight

In the **List View Highlight** and **Grid View Highlight** sections, you can specify properties for highlighting items in views.



LIST VIEW HIGHLIGHT

Range	NoHighlightRange
Move duration	-1
Move velocity	400
Resize duration	-1
Resize velocity	400


Preferred end 0

Follows current ✓ true

The current item in a list or grid view is higlighted if you set the value of the **Range** field to **ApplyRange** or **StrictlyEnforceRange**. When you select to apply the range, the view attempts to maintain the highlight within the range. However, the highlight can move outside of the range at the ends of the view or due to mouse interaction. When you select to enforce the range, the highlight never moves outside of the range. The current item changes if a keyboard or mouse action would cause the highlight to move outside of the range.

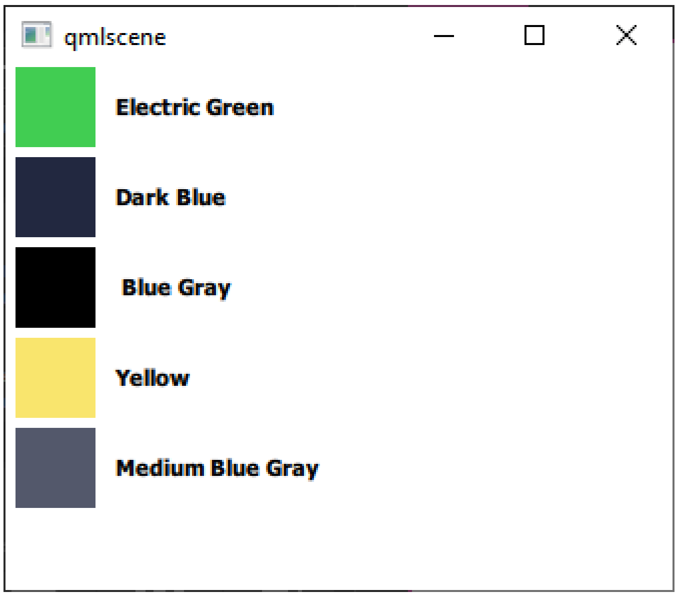
The values of the **Preferred begin** and **Preferred end** fields affect the position of the current item when the view is scrolled. For example, if the currently selected item should stay in the middle of the view when it is scrolled, set the begin and end values to the top and bottom coordinates of where the middle item would be. If the current item is changed programmatically, the view will automatically scroll so that the current item is in the middle of the view. The begin value must be less than the end value.

Select the **Follows current** check box to enable the view to manage the highlight. The highlight is moved smoothly to follow the current item. Otherwise, the highlight is not moved by the view, and any movement must be implemented by the highlight.

The values of the **Move duration**, **Move velocity**, **Resize duration**, and **Resize velocity** fields control the speed of the move and resize animations for the highlight.

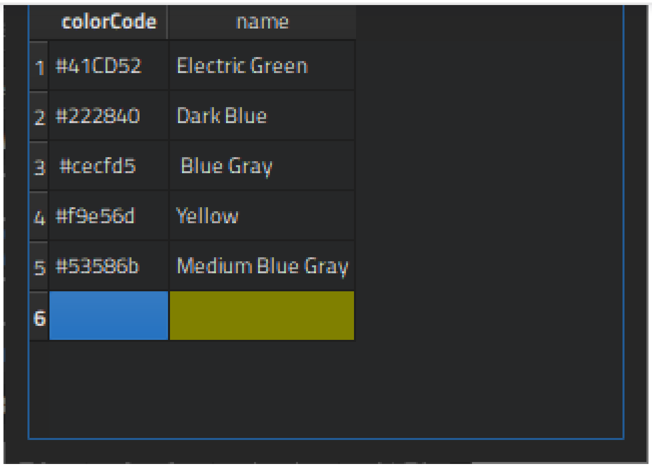
Editing List Models

When you add a Grid View, List View, or Path View, the ListModel and the delegate component that creates an instance for each item in the model are added automatically. For grid and list views, you can edit the list model in Qt Design Studio.



To edit list models:

1. Drag-and-drop a **Grid View** or **List View** from **Components** > **Default Components** > **Views** to the **Navigator** or **2D** view.
2. Right-click the view in **Navigator**, and select **Edit List Model** in the context-menu to open the list model editor:



	colorCode	name
1	#41CD52	Electric Green
2	#222840	Dark Blue
3	#cecfd5	Blue Gray
4	#f9e56d	Yellow
5	#53586b	Medium Blue Gray
6	🟦	🟨

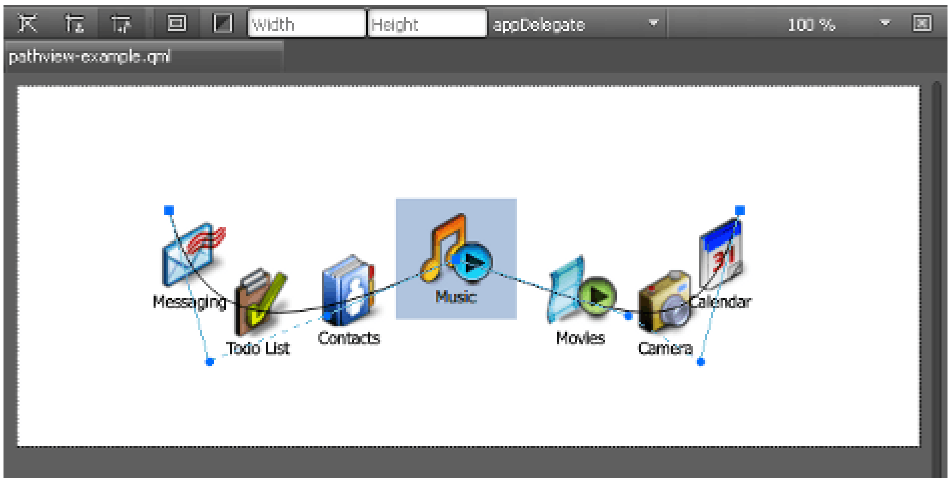
3. Double-click the column headings and cells to change their values.
4. Use the toolbar buttons to add, remove, or move rows and columns. In a list, each column represents a property and each row adds a list item.

You can replace the default model and delegate with other, more complex models and delegates in the Code view. Item Delegate and Swipe Delegate components are also available in **Components** > **Qt Quick Controls**.

Path View

A **Path View** component lays out data provided by data models on a Path.

A graphical spline editor enables you to specify path view paths, which is a non-trivial task to do in the **Code** view.



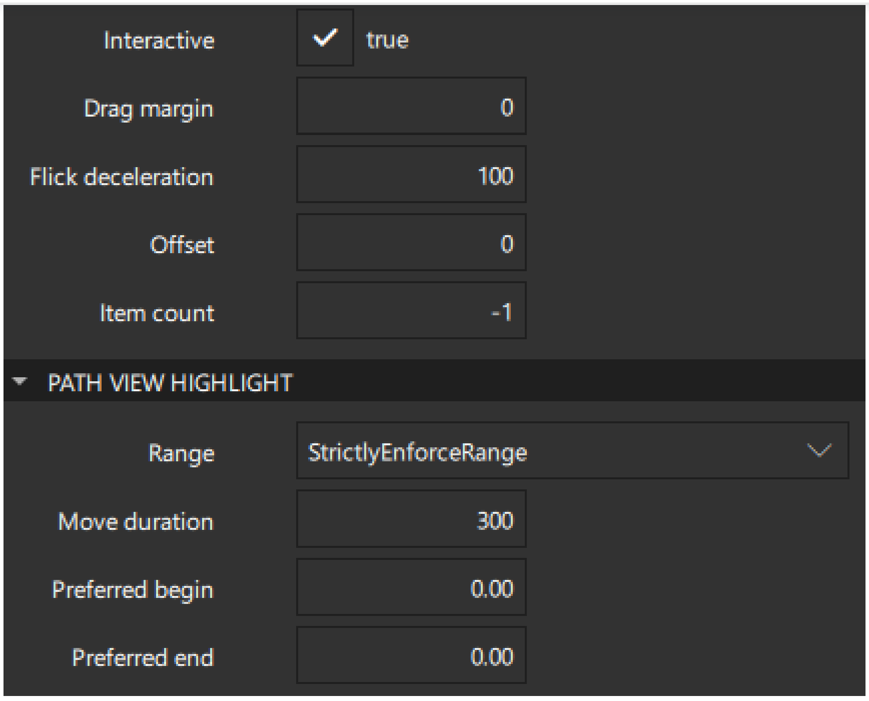
To start editing the path, double-click the path view in the 2D view. The editor composes the path of PathCubic path objects. They are cubic Bezier curves to a given position with two control points. Drag and drop the control points in the **2D** view to construct the curve.

In addition, PathLine and PathQuad path objects are supported indirectly. To make a curve segment linear, select **Make Curve Segment Straight** in the context menu.

By default, the path is closed, which means that its start and end points are identical. To create separate start and end points for it, right-click an edit point to open a context menu, and deselect **Closed Path**.

To add intermediary points to a curve segment, select **Split Segment** in the context menu.

In the **Path View** section, you can specify other properties for the path view. The value of the **Drag margin** field specifies the maximum distance from the path that initiates mouse dragging.



Interactive	✓ true
Drag margin	0
Flick deceleration	100
Offset	0
Item count	-1
PATH VIEW HIGHLIGHT	
Range	StrictlyEnforceRange
Move duration	300
Preferred begin	0.00
Preferred end	0.00

Select the **Interactive** check box to make an item flickable. The value of the **Flick deceleration** field specifies the rate at which a flick will decelerate.

In the **Offset** field, specify how far along the path the items are from their initial positions. This is a real number that ranges from 0 to the value of the **Item count** field, which displays the number of items in the model.

In the **Path View Highlight** section, you can specify properties for highlighting path objects.

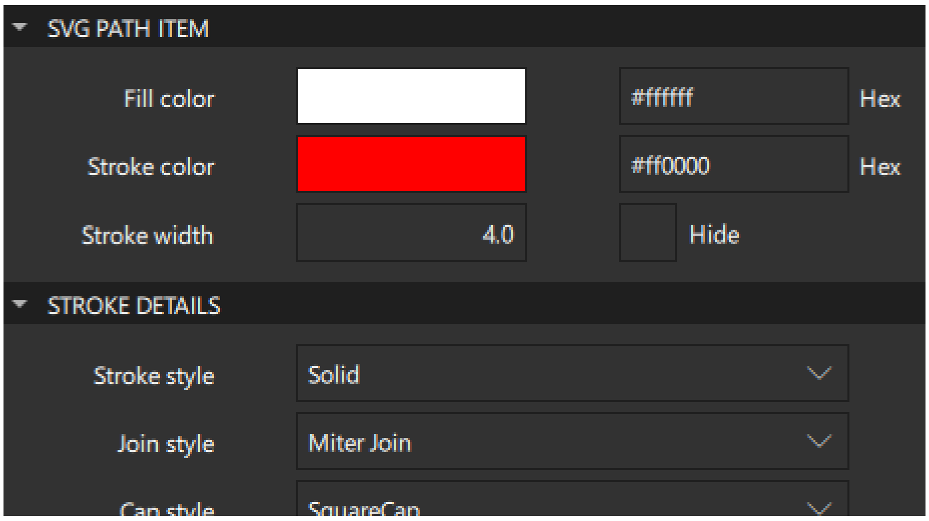
Note: You can also use the SVG Path Item Studio Component to specify an SVG path data string that draws a path.

SVG Path Item

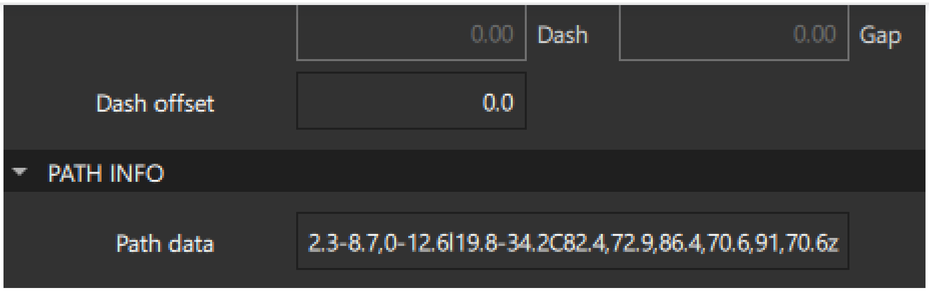
The **SVG Path Item** component uses an SVG path data string to draw a path as a line.

Setting path colors is described in Picking Colors.

The stroke property values that specify the appearance of the path are described in Strokes.



SVG PATH ITEM			
Fill color	⬜	#ffffff	Hex
Stroke color	🟥	#ff0000	Hex
Stroke width	4.0	☐ Hide	
STROKE DETAILS			
Stroke style	Solid		
Join style	Miter Join		
Cap style	SquareCap		



0.00 Dash 0.00 Gap

Dash offset 0.0

PATH INFO

Path data 2.3-8.7,0-12.6l19.8-34.2C82.4,72.9,86.4,70.6,91,70.6z

The **Path data** field in the **Path Info** section contains the SVG path data string that specifies the path. For more information, see W3C SVG Path Data.

Summary of Model Components

The following table lists the components that you can use to add data models to UIs. The *Location* column indicates the location of the component in **Components**. The *MCU* column indicates which components are supported on MCUs.

Icon	Name	Location	MCU	Purpose
	Grid View	Default Components - Views		A grid vizualization of a model.
	Item Delegate	Qt Quick Controls		A standard view item that can be used as a delegate in various views and controls, such as ListView and ComboBox.
	List View	Default Components - Views	✓	A list vizualization of a model.
	Path View	Default Components - Views		Vizualizes the contents of a model along a path.
	Scroll View	Qt Quick Controls		Provides scrolling for user-defined content. It can be used instead of a Flickable component.
	Stack View	Qt Quick Controls		A stack-based navigation model.
	SVG Path Item	Qt Quick Studio Components		An SVG path data string that is used to draw a path as a line.
	Swipe Delegate	Qt Quick Controls		A view item that can be swiped left or right to expose more options or information. It is used as a delegate in views such as ListView.
	Swipe View	Qt Quick Controls	✓	Enables users to navigate pages by swiping sideways.

< UI Controls 2D Effects >