建模

可以使用模型编辑器创建通用建模语言 （UML） 样式的模型，其中包含提供系统不同视图的结构化关系图和行为图。但是，编辑器使用 UML 的变体，并且仅提供属性子集来指定模型元素的外观。

结构图表示系统的静态方面，因此是稳定的，而行为图同时具有静态和动态方面。

可以创建以下类型的结构图：

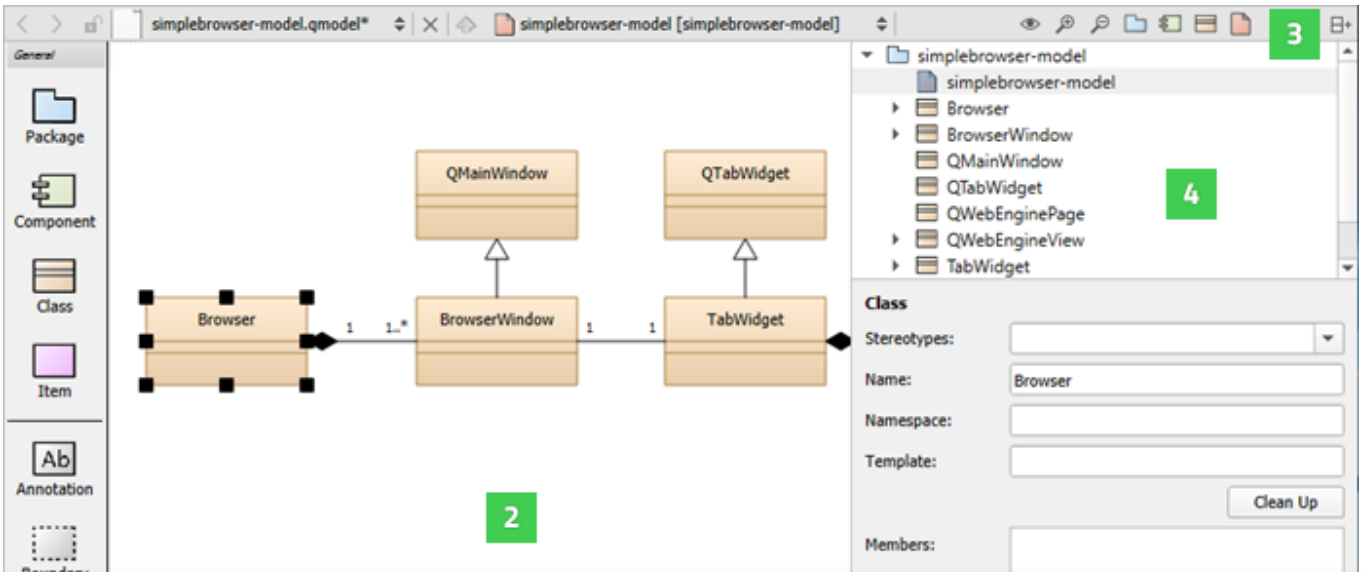
- 包图，由包及其关系组成，并可视化系统的打包方式。
- 类图，由类、依赖项、继承、关联、聚合和组合组成，并提供系统的面向对象视图。
- 组件图，表示一组组件及其关系，并提供系统的实现视图。
- 部署关系图，表示一组软件和硬件组件及其关系，并可视化系统的部署。

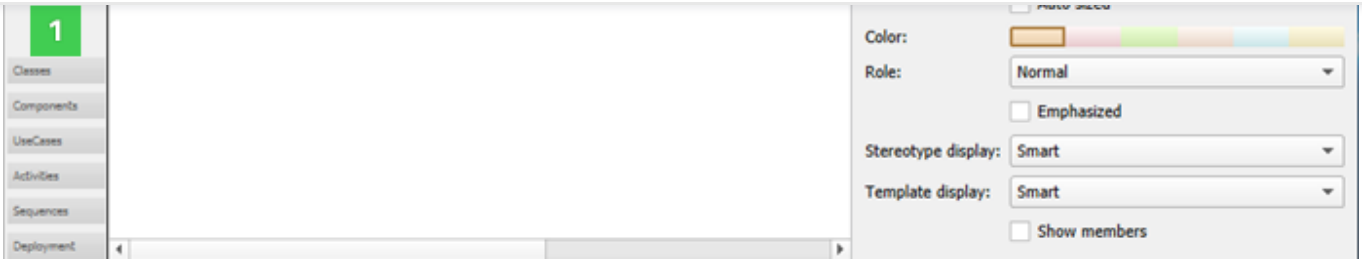
您可以创建以下类型的行为图：

- 用例图，由参与者、用例及其关系组成，表示系统的特定功能。
- 活动图，用于可视化从一个活动到另一个活动的流。
- 序列图，由实例组成，并指定实例的激活和销毁位置及其生命线的结束位置。

使用模型编辑器

您可以创建包含多个不同结构图或行为图的模型。可以向关系图中添加元素并为其指定属性。您可以使用标准模型元素，也可以使用自定义图标添加自己的元素。





可以通过以下方式将模型元素添加到关系图中：

- 将模型元素从元素工具栏 （1） 拖放到编辑器 （2）。
- 选择工具栏按钮 （3） 以将元素添加到元素树 （4）。
- 将元素从元素树拖到编辑器中，以将它们及其所有关系添加到图中。
- 将源文件从侧边栏视图拖放到编辑器中，以将C++类或组件添加到图中。

您可以通过用边界包围元素来对元素进行分组。移动边界时，其内的所有元素将一起移动。同样，将泳道拖到图中。当您移动泳道时，所有直至泳道（对于垂直泳道）或在其下方（对于水平泳道）的元素都将一起移动。当您将泳道图标放在图的上边框上时，将创建垂直泳道，当您将图标放在左边框附近时，将创建水平泳道。

放置在包上的类或其他对象将与包一起移动。您可以通过选择单个元素来移动它们并修改其属性 （5）。您还可以使用*多选*功能对元素进行临时分组。

要对齐编辑器中的元素，请选择多个元素，然后单击鼠标右键以打开上下文菜单。在"**对齐对象**"菜单中选择动作可水平或垂直对齐元素，或者调整其宽度和高度。

将鼠标拖到元素上以选择它们并应用诸如更改其*构造型*或颜色之类的操作。构造型是元素（如*实体*、*控件*、*接口*或*边界*）的分类器。实体通常是用于存储数据的类。对于某些构造型，定义了自定义图标。您可以将多个逗号分隔的构造型分配给一个元素。

若要向图中添加相关元素，请在编辑器中选择一个元素，然后在上下文菜单中选择"**添加相关元素**"。

默认情况下，当您选择关系图中的某个元素时，该元素也会在**"结构"**视图中突出显示。若要更改此行为，以便

在"**结构**"中选择某个元素，使其在关系图中也突出显示，请单击并按住该按钮，然后选择"**将关系图与结构同步"**。若要使关系图中的选择与**"结构"**视图保持同步，请选择"**保持同步"**。

To zoom into diagrams, select the **Zoom In** toolbar button, press **Ctrl++**, or press **Ctrl** and roll the mouse wheel up. To zoom out of diagrams, select **Zoom Out**, press **Ctrl+-**, or press **Ctrl** and roll the mouse wheel down. To reset the diagram size to 100%, select **Reset Zoom** or press **Ctrl+0**.

To print diagrams, press **Ctrl+C** when no elements are selected in the editor to copy all elements to the clipboard by using 300 dpi. Then paste the diagram to an application that can print images.

If you copy a selection of elements in the editor, only those elements and their relations will be copied to the clipboard as an image.

To save diagrams as images, select **File** > **Export Diagram**. To save only the selected parts of a diagram, select **Export Selected Elements**.

Creating Models

You can use wizards to create models and *scratch models*. A scratch model can be used to quickly put a temporary diagram together. The wizard creates the model file in a temporary folder without any input from you. Therefore, you can assign a keyboard shortcut to the wizard and use it to create and open models with empty diagrams.

To create models:

2. Drag and drop model elements to the editor and select them to specify properties for them.

    1. In the **Stereotypes** field, enter the stereotype to apply to the model element or select a predefined stereotype from the list.
    2. In the **Name** field, give a name to the model element.
    3. Select the **Auto sized** check box to reset the element to its default size after you have changed the element size by dragging its borders.
    4. In the **Color** field, select the color of the model element.
    5. In the **Role** field, select a *role* to make the model element color lighter, darker, or softer. You can also remove color and draw the element outline or flatten the element by removing gradients.
    6. Select the **Emphasized** check box to draw the model element with a thicker line.
    7. In the **Stereotype display** field, select:

        - **Smart** to display the stereotype as a **Label**, a **Decoration**, or an **Icon**, depending on the properties of the element. For example, if a class has the stereotype **interface**, it is displayed as an icon until it becomes displayed members, after which it is displayed as a decoration.
        - **None** to suppress the displaying of the stereotype.
        - **Label** to display the stereotype as a line of text using the standard form above the element name even if the stereotype defines a custom icon.
        - **Decoration** to show the standard form of the element with the stereotype as a small icon placed top right if the stereotype defines a custom icon.
        - **Icon** to display the element using the custom icon.

3. To create a relation between two elements, select the arrow icon next to an element and drag it to the end point of the relation.
4. Select the relation to specify settings for it, according to its type: inheritance, association, or dependency. You can specify the following settings for dependency relations, which are available for all element types:

    1. In the **Stereotypes** field, select the *stereotype* to apply to the relation.
    2. In the **Name** field, give a name to the relation.
    3. In the **Direction** field, you can change the direction of the connection or make it bidirectional.

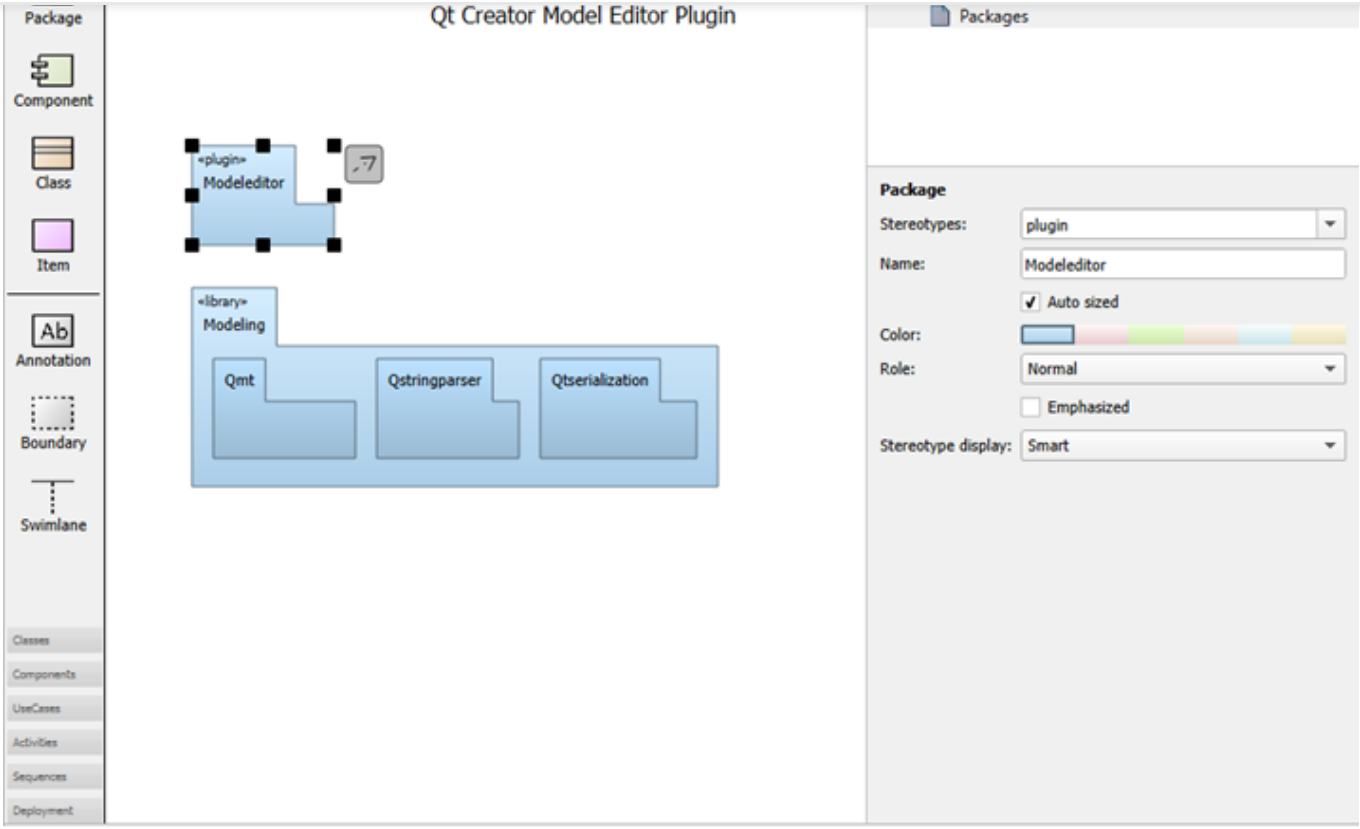
5. To move the end of a relation to a different element, grab the end point and drop it over another element that accepts relations of that type. For example, only classes accept inheritances and associations.
6. To create *sampling points* that divide a relation into two connected lines, select a relation and press **Shift** and click on the relation line.

    If possible, the end point of a relation is moved automatically to draw the line to the next sampling point either vertically or horizontally.

7. To remove a sampling point, press **Ctrl** and click the sampling point.
8. To group elements, drag and drop a **Boundary** element to the editor and resize it to enclose the elements in the group.

Creating Package Diagrams

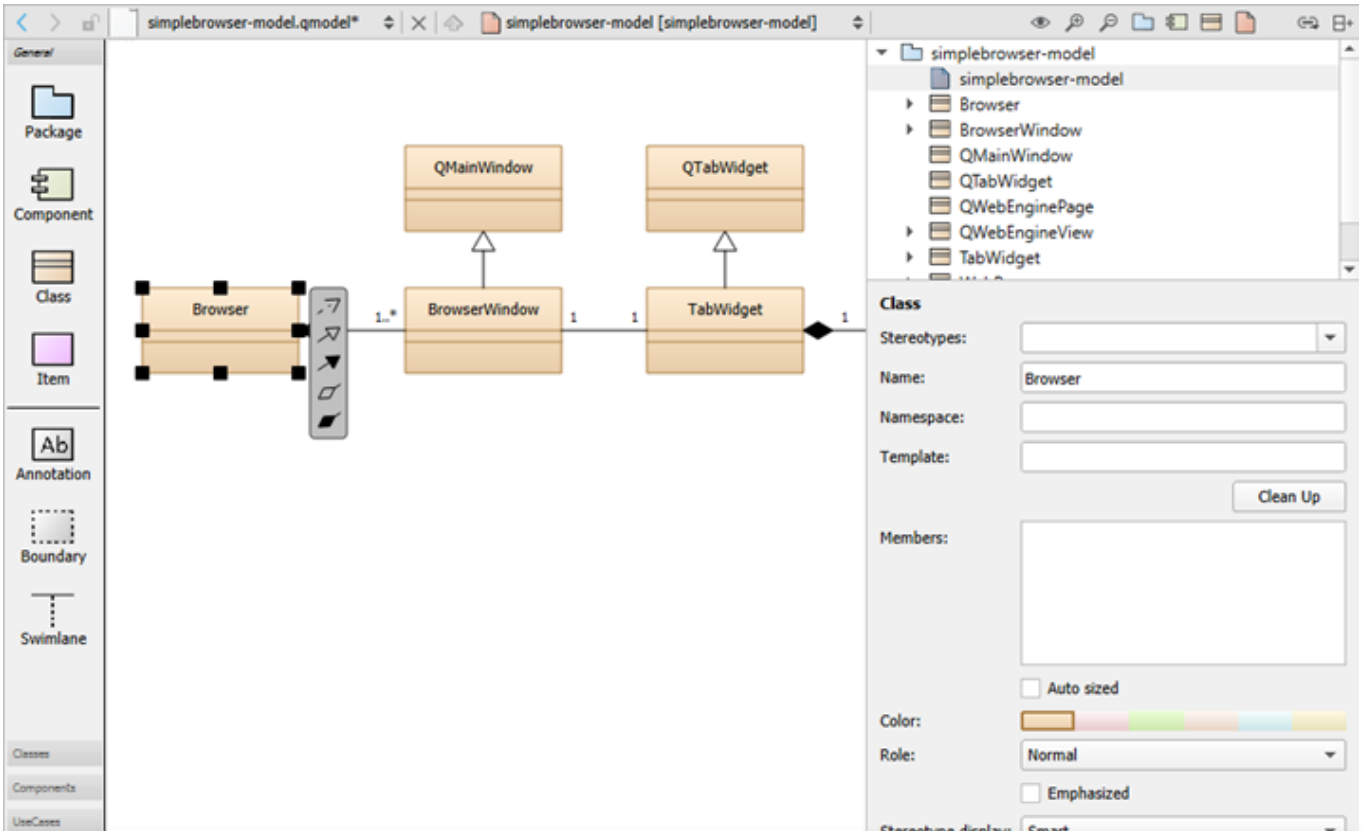
You can add nested package elements to a package diagram. The depth of the elements in the diagram corresponds to the depth of the structured model. Elements stacked on other elements of the same type are automatically drawn in a darker shade of the selected color.



Right-click a package to open a context menu, where you can select **Create Diagram** to create a new package diagram within the model. You can drag and drop items from the element tree to the diagram.

To update the include dependencies of the package, select **Update Include Dependencies**.

Creating Class Diagrams



To create class diagrams:

1. To add C++ classes to class diagrams, drag and drop files from **Projects** to the editor, and select **Add Class**.
2. In addition to the common element properties, you can specify the following properties:
   - In the **Template** field, specify the template to use.
   - In the **Template display** field, select the display format for the template:
     - **Smart** displays the template as **Box** or **Angle brackets**, depending on the class properties.
     - **Box** displays the template in a small box with a dotted border in the top right corner of the class icon.
     - **Angle brackets** writes the template in angle brackets behind the class name using the C++ syntax.
   - In the **Members** field, specify members for the class, as described in Specifying Members.
   - Select **Clean Up** to format the contents of the **Members** field depending on their visibility (private, protected, public) and following the rules set for whitespace, line breaks, and so on.
   - Select the **Show members** check box to show the members in the diagram.

To navigate from a class in a diagram to the source code, double-click the class in the editor or select **Show Definition** in the context menu.

Adding Relations

Elements in class diagrams can have the following types of relations: inheritance, association, and dependency. The end points of association relations can have the following properties: role, cardinality, navigable, and relationship.

To create self-relations, start creating a new association and press **Shift** to create a new *sampling point* while dragging the association. Create another sampling point and drop the association at the same class.

To add more points, press **Shift** and click a relation. To delete a point, press **Ctrl** and click a point.

Specifying Members

To specify members for the class, enter each member on a separate line using a C++ like syntax. For example, the following lines define the method that is private, virtual, and constant:m

```
private:
virtual int m(string a) const;
```

You may group members:

```
[Geometry]
QPointF position;
QSizeF size;
```

You may add stereotypes to members:

```
<<setter>> setPosition(const QPointF &pos);
```

There are some limitations of the parser:

- Multi-line declarations work only if lines are wrapped within nested brackets:

```
void setSize(int width,
             int height);
```

- Preprocessor macros will not be translated. Some Qt keywords are recognized (for example Q_SLOT).
- Function pointer declarations are interpreted as methods.
- `throw()` and specifiers are not ignored but will make the declaration a method.`noexpect()`

Creating Component Diagrams

You can add source code components, such as libraries, databases, programs, and architectural layers to a component diagram. To add components to component diagrams, drag and drop source code from **Projects** to the editor, and select **Add Component**.

To navigate from a component in a diagram to the source code, double-click the component in the editor or select **Show Definition** in the context menu.

Adding Custom Elements

The model editor provides the following built-in element types: package, component, class, and item. For package, component, and class elements, you can specify custom icons. The color, size, and form of the icon are determined by a stereotype. If you attach the stereotype to an element, the element icon is replaced by the custom icon. For example, you can attach the entity and interface stereotypes to classes and the database stereotype to components.

The use case and activity diagrams are examples of using the built-in *item* element type to add custom elements. The item element has the form of a simple rectangle. The use case illustrates how to use a custom icon for an item. The attached stereotype is called *usecase* but it is hidden. Therefore, if you drag the use case to the diagram, it is shown as a use case but no stereotype appears to be defined and you can attach an additional stereotype to the use case.

Color and icons are attached to elements in use case and activity diagrams by using a simple definition file format. For example, the following code adds the custom element:`UseCase`

```
Icon {
    id: UseCase
    title: "Use-Case"
    elements: item
    stereotype: 'usecase'
    display: icon
    width: 40
    height: 20
    baseColor: #5fb4f0
```

```
        }
    }
```

For more information about the available options, see *standard.def* in the *share/qtcreator/modeleditor* directory in the Qt Creator installation directory. It describes also how to define custom relation types and templates for existing types (such as a composition relation that can be drawn between classes).

You can add your own definition file and save it with the file extension *.def* to add custom colors and icons for stereotypes, elements, or tool bars. Either store this file in the the same directory as the *standard.def* file or select the root element of a model and apply your *.def* file to the property **Config path**.

‹ Editing MIME Types

Editing State Charts ›