# Qt 之布局管理器

# 简述

Qt的布局系统提供了一个简单的和强有力的方式，来自动排列窗口子控件布局。

所有QWidget子类可以使用布局来管理他们的子控件。QWidget::setLayout()函数可以为一个控件布局。当通过这种方式布局以后，它负责以下任务:

* 布置子控件。
* 最高层窗口可感知的默认大小。
* 最高层窗口可感知的最小大小。
* 调整大小的处理。
* 当内容改变的时候自动更新：   
  + 字体大小、文本或者子控件的其它内容。
  + 隐藏或者显示子控件。
  + 移除一些子控件。

# 

# Qt的布局类

Qt的布局类使用手写的C++代码设计的，所以很容易理解和使用。

使用Qt Designer创建的界面生成的代码也使用了布局类。涉及用户界面开发时，Qt Designer非常有用，因为它避免了编译、链接、运行这样一个循环。

| **类** | **描述** |
| --- | --- |
| QBoxLayout | 水平或垂直排列控件 |
| QButtonGroup | 组织按钮的容器 |
| QFormLayout | 管理输入控件和其相关的标签 |
| QGraphicsAnchor | 表示在QGraphicsAnchorLayout中两个项目之间的锚 |
| QGraphicsAnchorLayout | 在图形视图中可以将锚连接到一起 |
| QGridLayout | 网格布局（多行多列） |
| QGroupBox | 带标题的分组框 |
| QHBoxLayout | 水平排列控件 |
| QLayout | 几何管理器的基类 |
| QLayoutItem | 抽象的操作布局Item |
| QSizePolicy | 描述水平和垂直大小调整的策略 |
| QSpacerItem | 布局中的空间隔 |
| QStackedLayout | 切换控件，同一时间只有一个控件可见 |
| QStackedWidget | 切换控件，同一时间只有一个控件可见 |
| QVBoxLayout | 垂直排列控件 |
| QWidgetItem | 表示一个控件的布局项 |

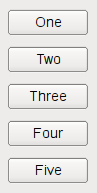
# 水平、垂直、网格、表单布局

为窗口提供一个好布局的最佳方式是使用内置的布局管理器：QHBoxLayout、QVBoxLayout、QGridLayout和QFormLayout。这些类都派生自QLayout，QLayout又派生自QObject（非QWidget），布局窗口自动地把子窗口按照它们被构造地顺序进行布局。要生成更复杂的布局，可以在其它布局里面嵌入另一个布局。

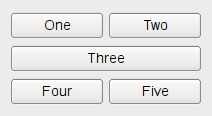
* QHBoxLayout：把子窗口从左到右排列在一个水平行上。

这里写图片描述

* QVBoxLayout：把子窗口从上到下排列在一个垂直列上。



* QGridLayout：把子窗口排列在一个二维的网格中，窗口可占据多个单元格。



* QFormLayout：把子窗口按照标签-输入框的形式排列在两列。



# 代码布局

下面的代码创建一个管理五个按钮的水平布局，上面的第一张图所示：

QWidget \*window = new QWidget;

QPushButton \*button1 = new QPushButton("One");

QPushButton \*button2 = new QPushButton("Two");

QPushButton \*button3 = new QPushButton("Three");

QPushButton \*button4 = new QPushButton("Four");

QPushButton \*button5 = new QPushButton("Five");

QHBoxLayout \*layout = new QHBoxLayout;

layout->addWidget(button1);

layout->addWidget(button2);

layout->addWidget(button3);

layout->addWidget(button4);

layout->addWidget(button5);

window->setLayout(layout);

window->show();

对于QVBoxLayout代码是相同的，除了QHBoxLayout 和QVBoxLayout区别以外。对于QGridLayout有点不同，因为需要指定子控件的行和列的位置：

QWidget \*window = new QWidget;

QPushButton \*button1 = new QPushButton("One");

QPushButton \*button2 = new QPushButton("Two");

QPushButton \*button3 = new QPushButton("Three");

QPushButton \*button4 = new QPushButton("Four");

QPushButton \*button5 = new QPushButton("Five");

QGridLayout \*layout = new QGridLayout;

layout->addWidget(button1, 0, 0);

layout->addWidget(button2, 0, 1);

layout->addWidget(button3, 1, 0, 1, 2);

layout->addWidget(button4, 2, 0);

layout->addWidget(button5, 2, 1);

window->setLayout(layout);

window->show();

第三个QPushButton占据2列。通过指定QGridLayout:: addWidget()的第五参数为2来实现。

QFormLayout将在一行上添加两个控件，通常是QLabel和QLineEdit。在同一行中添加QLabel和QLineEdit，将把QLineEdit设置为QLabel的伙伴（在一起。。。(⊙o⊙)）。下面的代码将使用QFormLayout将3个QPushButton和相应QLineEdit排列起来。

QWidget \*window = new QWidget;

QPushButton \*button1 = new QPushButton("One");

QLineEdit \*lineEdit1 = new QLineEdit();

QPushButton \*button2 = new QPushButton("Two");

QLineEdit \*lineEdit2 = new QLineEdit();

QPushButton \*button3 = new QPushButton("Three");

QLineEdit \*lineEdit3 = new QLineEdit();

QFormLayout \*layout = new QFormLayout;

layout->addRow(button1, lineEdit1);

layout->addRow(button2, lineEdit2);

layout->addRow(button3, lineEdit3);

window->setLayout(layout);

window->show();

**使用布局提示**

当使用布局的时候，构建子控件的时候不需要指定parent，布局将会自动的指定parent（使用QWidget::setParent()），使它们成为安装了该布局的界面的子控件。

**注意：**布局中的控件是安装了该布局的界面的子控件，而非布局自身的，控件只能以控件作为parent，不能是布局。

也可以在布局中使用addLayout()来添加布局，内部的布局就会变成它的子布局。

# 为布局添加控件

当添加一个控件到一个布局中，布局过程工作如下：

* 所有的控件将最初根据它们的 QWidget::sizePolicy()和QWidget::sizeHint()而被分配到一定空间中。
* 如果任何一个控件有一个伸展因素设置，而且数值大于零，那么它们就会被根据它们的伸展因素的比例分配空间。
* 如果任何一个控件有一个伸展因素设置而且数值为零，那么只有当其它控件不再需要空间的时候才会得到更多的空间。在这当中，空间会首先被根据延展大小策略分配给控件。
* 任何控件被分配的空间的大小如果小于它们的最小大小（或者是在没有规定最小大小时的最小大小的提示），它们就会被按它们所需要的最小大小分配空间。（如果控件的伸展因素是它们的决定因素的情况下，它们不必有最小大小或者最小大小的提示。）
* 任何控件被分配的空间的大小如果大于它们的最大大小，它们就会被按它们所需要的最大大小分配空间。（如果控件的伸展因素是它们的决定因素的情况下，它们不必有最大大小。）

## 伸展因素

控件通常是在没有伸展因素设置的情况下被生成的。当它们被布置到一个布局中时，控件会被根据它们的QWidget::sizePolicy()或者它们的最小大小的提示中大的那一个分配给整个空间的一部分。伸展因素是用来根据控件互相的比例来改变它们所被分配的空间。

如果你使用一个QHBoxLayout来布置没有伸展参数设置的三个控件，我们就会得到像下面这样的布局：

这里写图片描述

如果我们给每个控件设置一个伸展因素，它们就会被按比例布置（但是不能小于最小大小的提示），举例来说：

这里写图片描述

# 布局中自定义控件

当你创建自己的控件类时，也应该传递它的布局属性。如果这个控件有一个QLayout，这样的话就已经被处理了。如果这个控件不包括任何子控件，或者使用自定义布局，需要重新实现下面这些QWidget的成员函数：

* QWidget::sizeHint() 返回控件的优先选用的大小。
* QWidget::minimumSizeHint() 返回控件所能有的最小大小。
* QWidget::setSizePolicy() 指定控件所需要的空间。

只要大小提示、最小大小提示或者大小策略发生改变，都要调用QWidget::updateGeometry()。这会引起布局的重新计算。对updateGeometry()的多重调用只会引起一次重新计算。

如果你的控件的优先选用的高度依赖于它的实际宽度（比如一个自动断词的标签），在sizePolicy()中设置hasHeightForWidth()标记，并且重新实现QWidget::heightForWidth()。

即使你实现了heightForWidth()，提供一个好的sizeHint()仍然是必需的。

进一步实现，请参考：Trading Height for Width.。

# 手动布局

如果要生成一种特殊的布局，也可以按上面的描述来生成一个自定义控件。重新实现QWidget::resizeEvent()来计算所需要分配的大小并且给每一个子控件调用setGeometry()。

当布局需要重新计算的时候，控件会得到一个类型是QEvent::LayoutRequest的事件。重新实现被通知QEvent::LayoutRequest事件的QWidget::event()。

# 如何编写自定义布局管理器

手动布局另一种方法是通过继承QLayout类来实现自己的布局管理器。