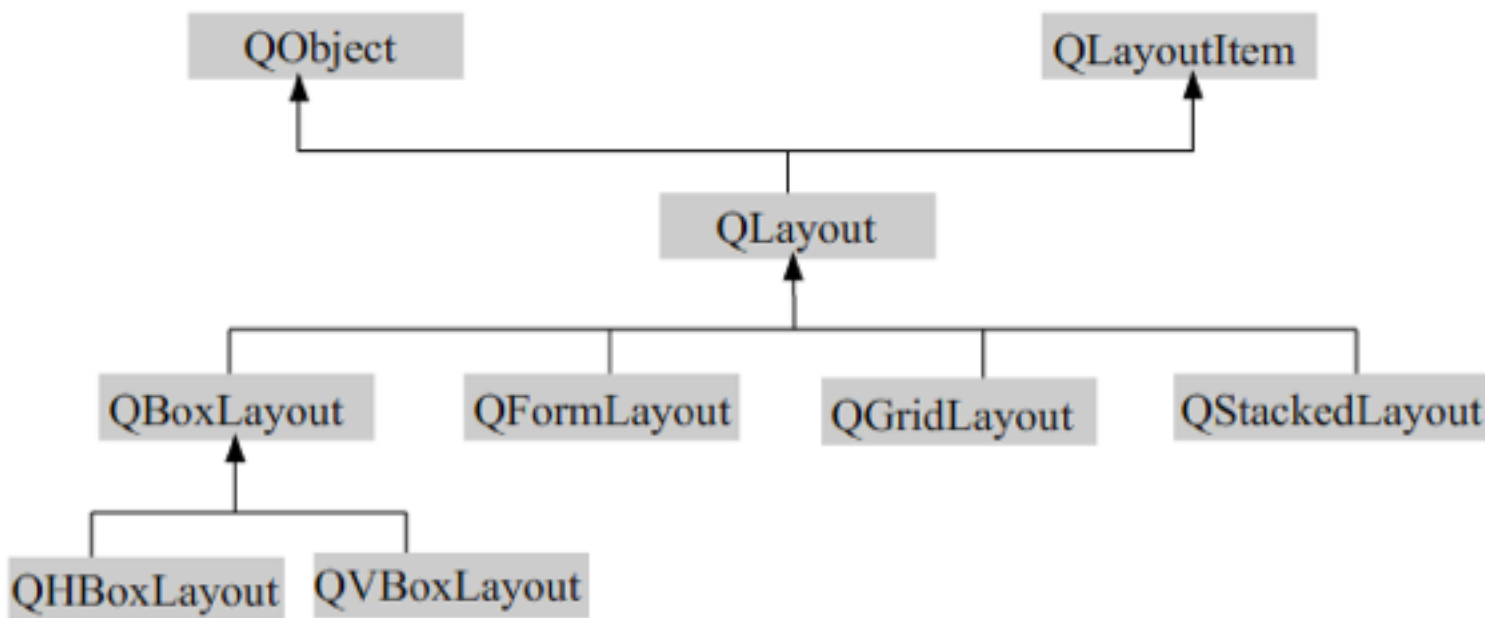


# 布局管理

南开大学软件学院

马玲 张圣林

对于一个完善的软件，布局管理却是必不可少的。无论是想要界面中部件有一个很整齐的排列，还是想要界面能适应窗口的大小变化，都要进行布局管理。Qt中主要提供了QLayout类及其子类来作为布局管理器，它们可以实现常用的布局管理功能



# 主 要 内 容

- 布局管理系统
- 设置伙伴
- 设置Tab键顺序
- 小结

# 布局管理系统

Qt的布局管理系统提供了简单而强大的机制来自动排列一个窗口中的部件，确保它们有效的使用空间。Qt包含了一组布局管理类来描述怎样在应用程序的用户界面对部件进行的布局，比如QLayout的几个子类，我们这里将它们称作布局管理器。所有的QWidget类的子类的实例（对象）都可以使用布局管理器来管理位于它们之中的子部件，QWidget::setLayout()函数可以在一个部件上应用布局管理器。一旦一个部件上设置了布局管理器，那么它会完成以下几种任务：

- 定位子部件；
- 感知窗口默认大小；
- 感知窗口最小大小；
- 改变大小处理；
- 当内容改变时自动更新：
  - 字体大小，文本或子部件的其他内容随之改变；
  - 隐藏或显示子部件；
  - 移除一个子部件。



# 布局管理器

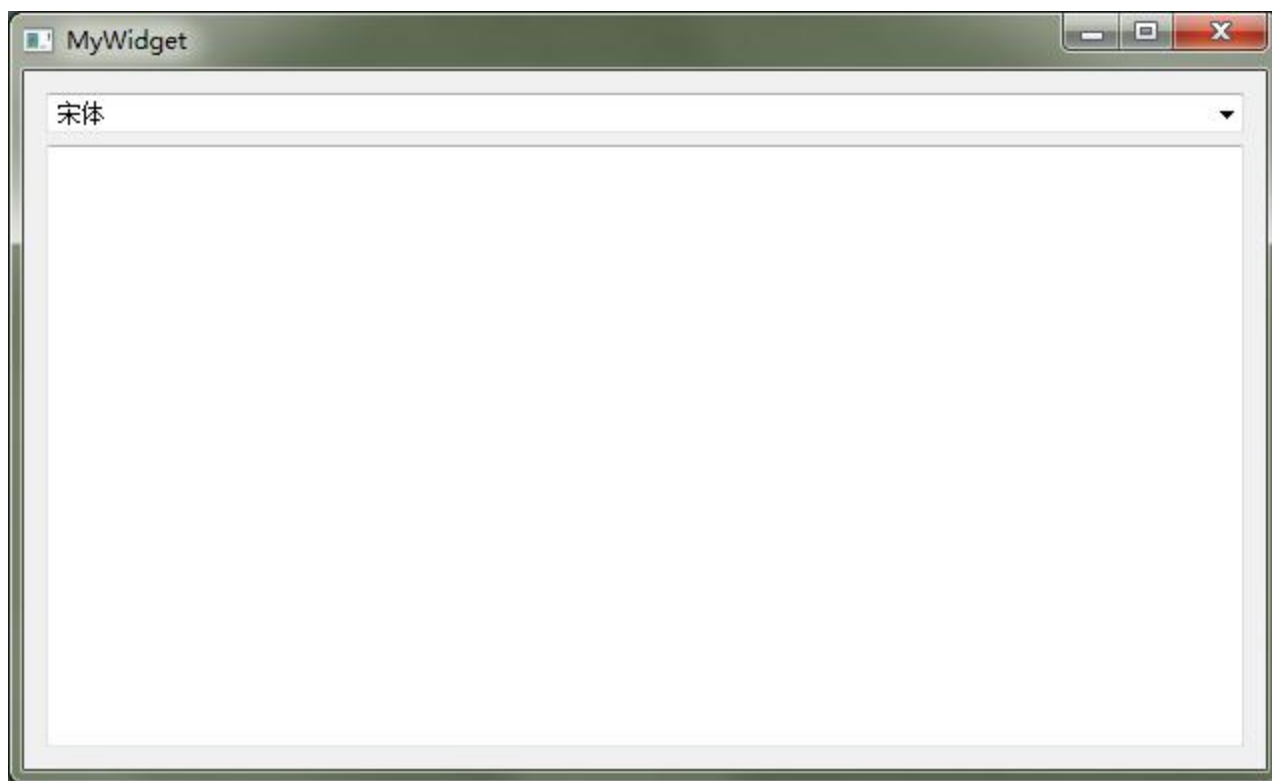
QLayout类是布局管理器的基类，它是一个抽象基类。该类继承自QObject和QLayoutItem类，而QLayoutItem类提供了一个供QLayout操作的抽象项目。QLayout和QLayoutItem都是在设计自己的布局管理器时才使用的，一般只需要使用QLayout的几个子类就可以了，它们分别是：

- QBoxLayout（基本布局管理器）
- QGridLayout（栅格布局管理器）
- QFormLayout（窗体布局管理器）
- QStackedLayout（栈布局管理器）



# 部件随窗口变化大小

在设计模式中向界面上拖入一个字体选择框Font Combo Box和一个文本编辑器Text Edit部件。然后点击主界面，并按下Ctrl+L快捷键，这样便设置了垂直布局管理器，可以看到两个部件已经填满了整个界面。这时运行程序，然后拉伸窗口，两个部件会随着窗口的大小变化而变化，这就是布局管理器在起作用。



# 基本布局管理器（QBoxLayout）

基本布局管理器QBoxLayout类可以使子部件在水平方向或者垂直方向排成一行，它将所有的空间分成一行盒子，然后将每个部件放入一个盒子中。它有两个子类QHBoxLayout水平布局管理器和QVBoxLayout垂直布局管理器。布局管理器的几个属性如下表所示。

| 属性                   | 说明               |
|----------------------|------------------|
| layoutName           | 现在所使用的布局管理器的名称   |
| layoutLeftMargin     | 设置布局管理器到界面左边界的距离 |
| layoutTopMargin      | 设置布局管理器到界面上边界的距离 |
| layoutRightMargin    | 设置布局管理器到界面右边界的距离 |
| layoutBottomMargin   | 设置布局管理器到界面下边界的距离 |
| layoutSpacing        | 布局管理器中各个子部件间的距离  |
| layoutStretch        | 伸缩因子             |
| layoutSizeConstraint | 设置大小约束条件         |



# 使用代码实现水平布局

```
QHBoxLayout *layout = new QHBoxLayout;           // 新建水平布局管理器

layout->addWidget(ui->fontComboBox);             // 向布局管理器中添加部件

layout->addWidget(ui->textEdit);

layout->setSpacing(50);                           // 设置部件间的间隔

layout->setContentsMargins(0, 0, 50, 100); // 设置布局管理器到边界的距离，
                                           // 四个参数顺序是左，上，右，下

setLayout(layout);
```





# 栅格布局管理器 (QGridLayout)

栅格布局管理器QGridLayout类使得部件在网格中进行布局，它将所有的空间分隔成一些行和列。行和列的交叉处就形成了单元格。然后将部件放入一个确定的单元。

QGridLayout

```
// 添加部件
```

```
layout->add
```

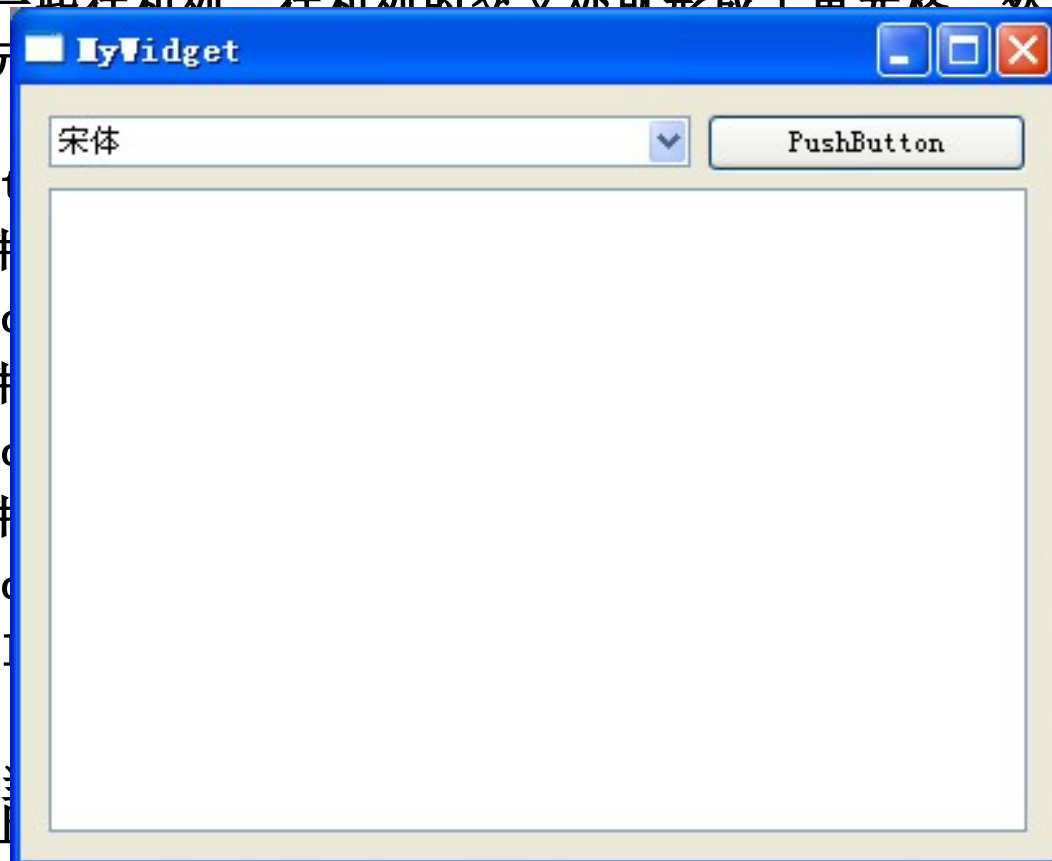
```
// 添加部件
```

```
layout->add
```

```
// 添加部件
```

```
layout->add
```

```
setLayout(1
```



**说明：**

窗口部件上

的父对象 (parent)

为这个窗口部件，所以在创建布局管理器

并不用指定父部件。

管理器再放到一个

自动重新定义自己

和其中的部件时



# 窗体布局管理器（QFormLayout）

窗体布局管理器QFormLayout类用来管理表单的输入部件和与它们相关的标签。窗体布局管理器将窗体中的部件分为两列，比如行编辑器和

在该界面“添加行”自动等，

The image shows a dialog box titled "添加表单布局行" (Add Form Layout Row) with a blue header bar. The dialog contains the following fields and controls:

- 标签文字 (L): 姓名 (aN):
- 标签名称 (N): nLabel
- 字段类型 (T): QLineEdit (selected from a dropdown menu)
- 字段名称 (F): nLineEdit
- 伙伴 (B): ☒
- 行 (R): 1 (with up/down arrows)

At the bottom of the dialog are two buttons: "确定" (OK) and "取消" (Cancel). The dialog is overlaid on a light gray grid background. To the left of the dialog, there is a small blue box with the text "宋体" (Songti) and a red line with square markers indicating a layout structure.

其拖入到表单布局下面便没名称”



这里填写的标签文字中 (&N)，要注意括号必须是英语半角的，表明它的快捷键是Alt+N，设置伙伴关系表示当按下Alt+N时，光标会自动跳转到标签后面对应的行编辑器中。按下确定键，便会在布局管理器中添加一个标签和一个行编辑器。按照这种方法，再添加三行：性别 (&S)，使用QComboBox；年龄 (&A)，使用QSpinBox；邮箱 (&M)，使用QLineEdit。



The image shows a Qt widget window titled "MyWidget". Inside the window, there is a form with the following elements:

- A label "姓名 (&N):" followed by a text input field.
- A label "性别 (&S):" followed by a QComboBox widget.
- A label "年龄 (&A):" followed by a QSpinBox widget showing the value "0".
- A label "邮箱 (&M):" followed by a text input field.
- A label "宋体" followed by a font family dropdown menu.
- A "PushButton" button.
- A large text area (multiline editor) at the bottom.

可以按下快捷键Alt

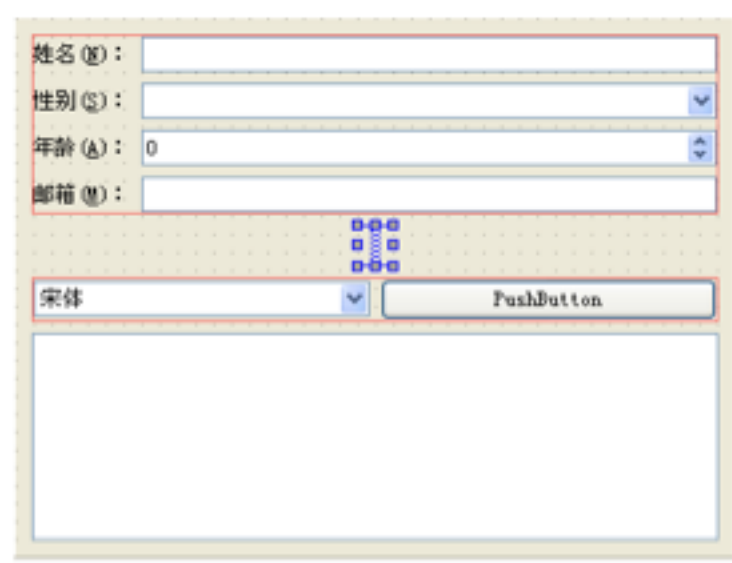
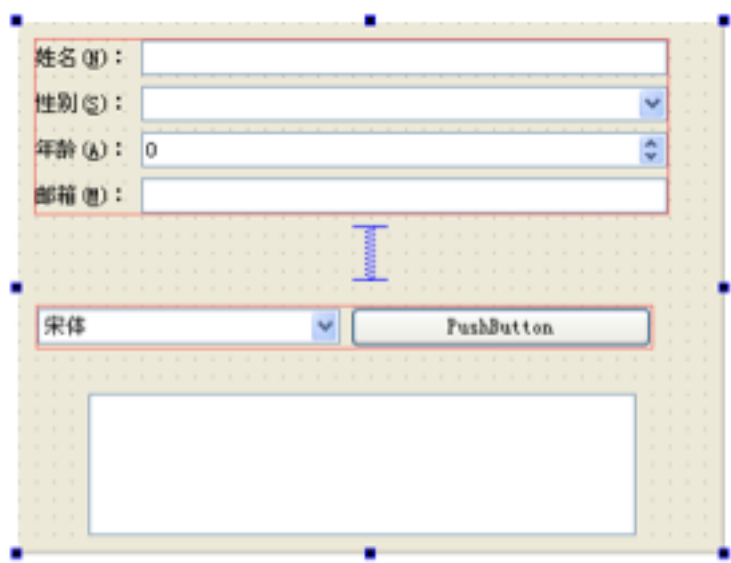
行编辑器中。



# 综合使用布局管理器

将前面的界面再进行设计：按下Ctrl键同时选中界面上的字体选择框fontComboBox和按钮pushButton，然后按下Ctrl+H快捷键将它们放入一个水平布局管理器中。然后再从部件栏中拖入一个Vertical Spacer垂直分隔符，它们是用来在部件间产生间隔的，将它放在表单布局管理器与水平布局管理器之间。

最后点击主界面，然后按下Ctrl+L快捷键，让整个界面处于一个垂直布局管理器中。这时我们可以在右上角的对象列表中选择分隔符Spacer，然后在属性栏中设置它的高度为100。



# 设置部件大小

凡是继承自QWidget的类都有这两个属性：大小提示（sizeHint）和最小大小提示（minimumSizeHint）。

- sizeHint属性保存了部件的建议大小，对于不同的部件，默认拥有不同的sizeHint，程序中使用sizeHint()函数来获取sizeHint的值；
- minimumSizeHint保存了一个建议的最小大小。程序中使用minimumSizeHint()函数来获取minimumSizeHint的值。

需要说明的是，如果使用minimumSize()函数设置了部件的最小大小，那么最小大小提示将会被忽略。



# 大小策略（sizePolicy）属性

sizePolicy属性的所有取值如下表所示：

| 常量                            | 描述  |
|-------------------------------|---|
| QSizePolicy::Fixed            | 只能使用 sizeHint() 提供的值，无法伸缩                       |
| QSizePolicy::Minimum          | sizeHint() 提供的大小是最小的，部件可以被拉伸                    |
| QSizePolicy::Maximum          | sizeHint() 提供的是最大大小，部件可以被压缩                     |
| QSizePolicy::Preferred        | sizeHint() 提供的大小是最佳大小，部件可以被压缩或拉伸                |
| QSizePolicy::Expanding        | sizeHint() 提供的是合适的大小，部件可以被压缩，不过它更倾向于被拉伸来获得更多的空间 |
| QSizePolicy::MinimumExpanding | sizeHint() 提供的大小是最小的，部件倾向于被拉伸来获取更多的空间           |
| QSizePolicy::Ignored          | sizeHint() 的值被忽略，部件将尽可能的被拉伸来获取更多的空间             |



# 伸缩因子（stretch factor）







它是用来设置部件间的比例的。

例如，界面上的字体选择框和一个按钮处于一个水平布局管理器中，现在想让它们的宽度比例为2:1，那么就可以点击对象栏中的horizontalLayout水平布局管理器对象，然后在它的属性栏中将layoutStretch属性设置为“2, 1”，这样这个水平布局管理器中的两个部件的宽度就是2:1的比例了。

如果要在代码中进行设置，可以在使用布局管理器的addWidget()函数添加部件的同时，在第二个参数中指定伸缩因子。



# QWidget部件大小相关属性

|   |                              |
|---|------------------------------|
|  geometry        | [ (0, 0), 400 x 300]         |
| X   | 0                            |
| Y   | 0                            |
| 宽度  | 400                          |
| 高度  | 300                          |
|  sizePolicy      | [Preferred, Preferred, 0, 0] |
| 水平策略  | Preferred                    |
| 垂直策略  | Preferred                    |
| 水平伸展  | 0                            |
| 垂直伸展  | 0                            |
|  minimumSize     | 0 x 0                        |
| 宽度  | 0                            |
| 高度  | 0                            |
|  maximumSize     | 16777215 x 16777215          |
| 宽度  | 16777215                     |
| 高度  | 16777215                     |
|  sizeIncrement | 0 x 0                        |
| 宽度  | 0                            |
| 高度  | 0                            |
|  baseSize      | 0 x 0                        |
| 宽度  | 0                            |
| 高度  | 0                            |

- 高度与宽度属性，是现在界面的大小；
- sizePolicy属性可以设置大小策略以及伸缩因子；
- minimumSize属性用来设置最小大小；
- maximumSize属性设置最大大小；
- sizeIncrement属性和baseSize属性是设置窗口改变大小的，一般不用设置。





# 可扩展窗口

对于一个窗口，可能有很多选项是扩充的，只有在必要的时候才显示出来，这就是所谓的特性，那新显示时，

首先在界面

然后转到它的

```
void MyWidget
```

```
{
```

```
    ui->textL
```

```
    if(checke
```

```
    else ui->
```

```
}
```



“显示可扩展窗

显示窗口按钮

器件的显示和隐藏

展窗口”));

);



使用按钮的按下与否两种状态来设置文本编辑器是否显示，并且相应的更改按钮的文本。为了让文本编辑器在一开始是隐藏的，还要在MyWidget类的构造函数中添加一行代码：

`ui->textEdit->hide();` // 让文本编辑器隐藏，也可以使用`setVisible(false)`函数

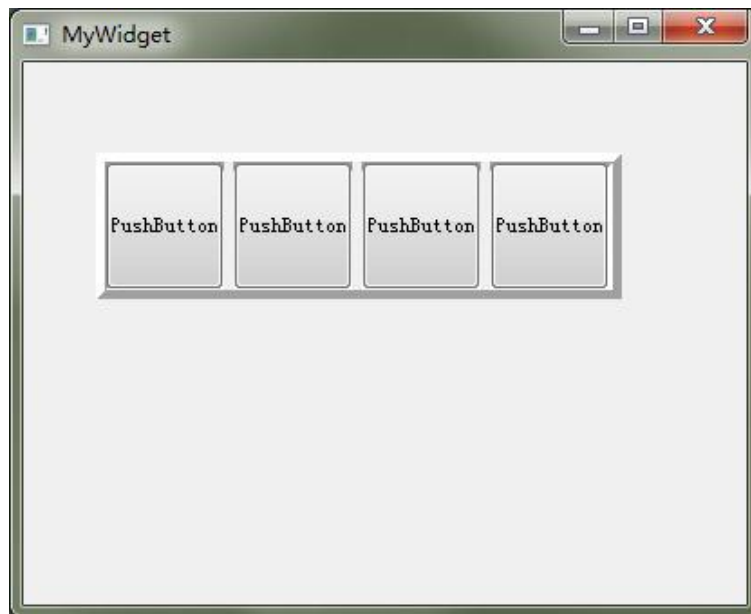


# 分裂器(QSplitter)

分裂器QSplitter类提供了一个分裂器部件。其实它和QBoxLayout很相似，可以完成布局管理器的功能，但是包含在它里面的部件，默认是可以随着分裂器的大小变化而进行相应大小变化的。

比如一个按钮，放在布局管理器中，它的垂直方向默认是不会被拉伸的，但是放到分裂器中就可以被拉伸。

还有一个不同就是，布局管理器是继承自QObject类的，而分裂器却是继承自QFrame类，QFrame类又是继承自QWidget类，也就是说，分裂器拥有QWidget类的特性，它是可见的，而且可以像QFrame一样设置边框。



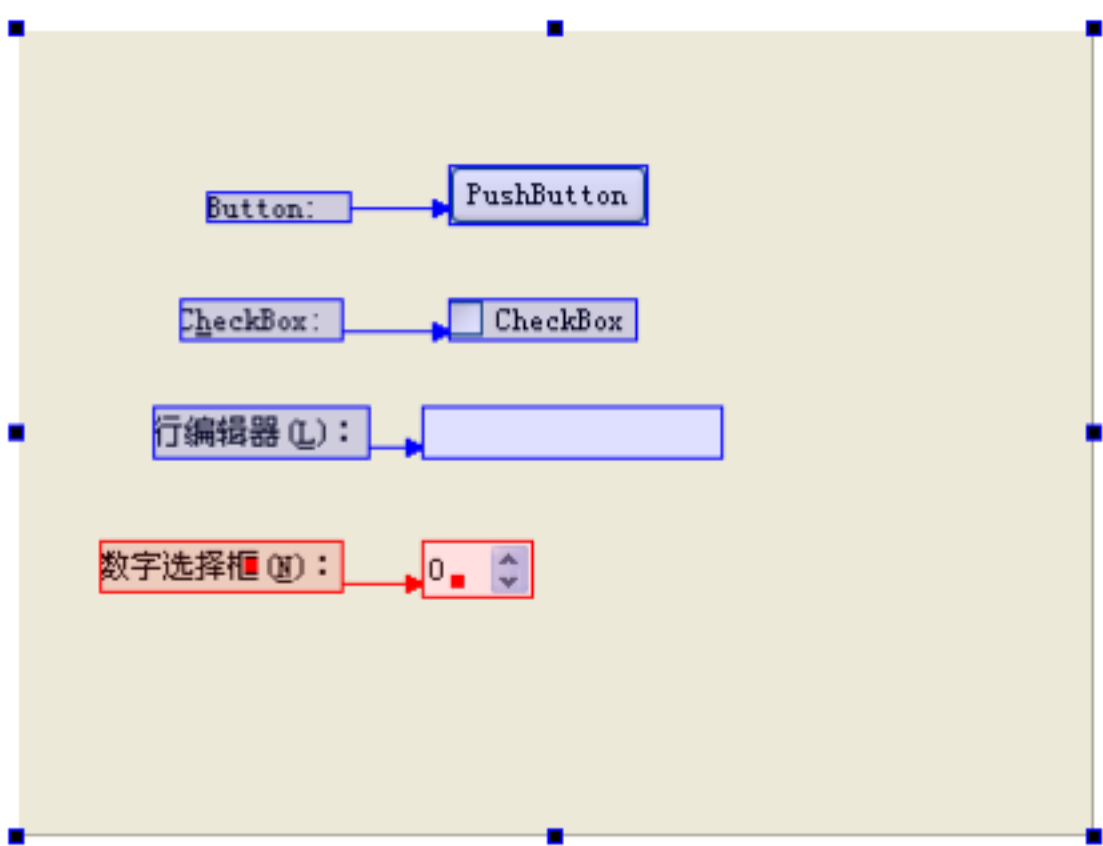
# 设置伙伴

- 伙伴 (buddy) 是在QLabel类中提出的一个概念。因为一个标签经常用作一个交互式部件的说明，就像表单布局管理器时看到的那样，一个lineEdit部件前面有一个标签说明这个lineEdit的作用。为了方便定位，QLabel提供了一个有用的机制，那就是提供了助记符来设置键盘焦点到对应的部件上，而这个部件就叫做这个QLabel的伙伴。
- 其中助记符就是我们所说的加速键。在使用英文标签时，在字符串的一个字母前面添加“&”符号，那么就可以指定这个标签的加速键是Alt加上这个字母，而对于中文，需要在小括号中指定加速键字母。



# 在设计器中设置伙伴

按下设计器顶部栏中的编辑伙伴图标，进入伙伴设计模式，分别将各个标签与它们后面的部件连起来。



# 设置Tab键顺序

对于一个应用程序，我们有时总希望使用Tab键来将焦点从一个部件移动到下一个部件。在设计模式，设计器提供了Tab键的设置功能。在上面程序的设计模式中，按下上边栏的编辑Tab顺序按钮进入编辑Tab键顺序模式，这时已经显示出了各个部件的Tab键顺序，只需要用鼠标点击这些数字，就可以更改它们。



## 使用代码进行设置

当程序启动时，焦点会在Tab键顺序为1的部件上。这里进行的设置等价于在构造函数中使用如下代码：

```
//lineEdit在spinBox前面
```

```
setTabOrder(ui->lineEdit, ui->spinBox);
```

```
//spinBox在pushButton前面
```

```
setTabOrder(ui->spinBox, ui->pushButton);
```

```
//pushButton在checkBox前面
```

```
setTabOrder(ui->pushButton, ui->checkBox);
```



## 小结

本章中讲述了一些关于界面布局的知识，学生要掌握几个布局管理器的使用方法，学会综合应用它们。

